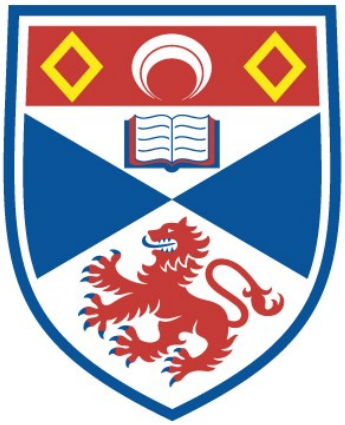
A UNIFIED STATE VARIABLE ANALYSIS OF REPETITIVELY PULSED RECOMBINATION LASERS

Andrew Keith Kidd

A Thesis Submitted for the Degree of PhD
at the
University of St Andrews



1992

A UNIFIED STATE VARIABLE ANALYSIS OF REPETITIVELY PULSED RECOMBINATION LASERS

A thesis presented by

Andrew Keith Kidd, B. Sc., M. Sc.,

to the

University of St. Andrews

in application for the degree of

Doctor of Philosophy

12th November, 1991

DECLARATION

I, Andrew Keith Kidd, hereby certify that this thesis has been composed by myself, that it is a record of my own work, and that it has not been accepted in partial or complete fulfilment of any other degree or professional qualification.

I was admitted to the Faculty of Science of the University of St. Andrews under Ordinance General No. 12 in September 1984 and as a candidate for the degree of Ph.D. in September 1985.

12th November, 1991

CERTIFICATE

I certify that Andrew Keith Kidd has spent nine terms of research work in the Department of Physics and Astronomy of St. Salvator's College, in the University of St. Andrews, under my supervision, that he has fulfilled the conditions of Ordinance No. 16 (St. Andrews), and that he is qualified to submit the following thesis in application for the degree of Doctor of Philosophy.

Arthur Maitland
Department of Physics and Astronomy
University of St. Andrews
12th November, 1991

AUTHOR'S CAREER

Andrew Keith Kidd was born in Wegberg, West Germany in 1961 and received his primary and secondary education in England and Wales. He obtained a joint honours degree in Physics and Electronics (1984) and the degree of M.Sc. in Optoelectronic and Laser Devices (1985) from the University of St. Andrews. In September 1985, he joined EEV Co. Ltd., Chelmsford as an Engineer seconded to the Department of Physics and Astronomy at the University of St. Andrews to conduct research associated with the content of this thesis.

ACKNOWLEDGEMENTS

I would like to acknowledge the supervision of Arthur Maitland. I am endebted to EEV Co. Ltd. for financial support and in particular to Colin Pirrie, Chris Neale and Hugh Menown. I am grateful to Laser One colleagues past and present involved directly with metal vapour laser research (Ewan, Graeme, Martin and Chris) for advice and assistance, to George 'Lou' Armstrong of the electronics workshop for persistance in the construction of the switched-mode power supply and to the participants in the five-a-side football matches every Monday and Wednesday lunchtime. Finally, this thesis is dedicated to my wife Karen and daughter Gail for their respective patience and peaceful nights during its completion.

Thesis title: **A Unified State Variable Analysis of Repetitively Pulsed Recombination Lasers**

DECLARATION OF UNRESTRICTED COPYRIGHT

In submitting this thesis to the University of St. Andrews I understand that I am giving permission for it to be made available for use in accordance with the regulations of the University Library for the time being in force, subject to any copyright vested in the work not being affected thereby. I also understand that the title and abstract will be published, and that a copy of the work may be made and supplied to any bona fide library or research worker.

25th March, 1992

A UNIFIED STATE VARIABLE ANALYSIS OF REPETITIVELY PULSED RECOMBINATION LASERS

ABSTRACT

A unified state-space model of a high repetition rate electrical discharge in a helium-strontium mixture is presented. The atomic (number density of particles in a particular state), optical (number density of photons), thermodynamic (local particle temperature and pressure) and electrical (voltage and current) internal states of the system are the state variables. A generalised circuit analysis program (GCAP) provides a description of the excitation circuit used to power the strontium laser. Rate equations describing the time evolution of the state variables are simultaneously numerically integrated to provide a description of the laser system during the discharge of a capacitor through the laser load and in the immediate afterglow.

The effects of parametric variation of the circuit on strontium laser performance are examined. The model predicts that Sr^{++} ions are formed during the discharge current pulse by step-wise excitation from SrII states lying lower in energy. A population inversion is achieved on the $6^2S_{1/2}$-$5^2P_{3/2}$ transition in SrII (λ=430.5nm) in the current pulse afterglow by rapid three-body recombination of Sr^{++} ions. A strong recombination flux is established on rapid termination of the discharge current pulse. Impedance matching of excitation circuit to load is essential for stimulated emission under recombination conditions.

The results of the state variable analysis are validated by experiment. The operating characteristics of a discharge-heated strontium vapour laser are presented. An average power of 0.3W is obtained in a high heat-loss configuration. The fall-time of the discharge current pulse is reduced by means of a saturable inductor placed in parallel with the laser load. The circuit generates current pulses with peak amplitudes up to 300A and fall-times of less than 70ns. GCAP is used to estimate the theoretical limits on the fall-time of the laser current by state variable analysis of a flux-controlled model of the saturable inductor.

A UNIFIED STATE VARIABLE ANALYSIS OF REPETITIVELY PULSED RECOMBINATION LASERS

Contents

Page

Chapter 1

Introduction

1.1 INTRODUCTION

Strontium and calcium vapour lasers (SVL and CaVL, respectively) represent valuable sources of high peak and potentially high average power radiation in the blue (λ = 430.5nm) and near UV (λ = 373.7nm) regions of the spectrum. They operate by discharging a capacitor at high repetition rates (100Hz - 30kHz) through a high-pressure (200mbar - 1atm.) helium buffer gas containing small pieces of the pure metal via either a longitudinal or transverse electrode configuration. This heats the laser cavity to a temperature sufficient to produce a suitable vapour pressure of the metal for lasing (~ 0.05mbar). As the optimum lasing temperature for strontium is approximately 600^{o}C, and that for calcium is only 100^{o}C higher, there is no need for multilayer insulation to obtain and maintain these temperatures. Advantages of these low temperature lasers include fast start-up, low technology designs and the possibility of producing sealed-off devices. The SVL operates with a conversion efficiency of about 0.1-0.14%, typically.

A large population of Sr^{++} ions is created during the current pulse by electron-impact. Strontium and calcium lasers are both examples of recombination lasers in which the population inversion is not achieved during the discharge current pulse risetime, but rather by the three-body recombination (e^{-}-e^{-}-Sr^{++}) of multiply-ionised species as they cascade through the excited states to the ground state in the discharge afterglow (Figure 1.1). The rate of this recombination process is highly temperature dependent (proportional to

$T_e^{-9/2}$, where T_e is the electron temperature[1]). Recombination lasers are run at high buffer gas pressures to cool the electrons via collisions with buffer gas atoms. Rapid termination of the current pulse applied to the laser head enhances electron cooling and allows recombination to proceed.

Among the advantages of non-equilibrium generated stimulated emission are the relatively long duration of the pulse (~200ns) and the short recovery time, the latter offering the possibility of high values of pulse repetition rate. As a research tool, the strontium laser is thus ideally suited to pumping blue laser dyes whose absorption maxima lie outside the scope of longer wavelength pulsed laser sources. Combination with the green and red lines of the copper and gold vapour lasers in a single device would provide a high-power "white-light" laser[2]. In the medical field, pulsed ultraviolet lasers appear to have several advantages over longer wavelength lasers[3]. The blue line is applicable to photodynamic therapy of neo-natal jaundice, where present techniques necessarily introduce harmful UV radiation. The use of a monochromatic source avoids this hazard[4].

1.2 RECOMBINATION LASERS

Stimulated emission in solid, liquid and gaseous media is well-established. However, there are two distinct types of lasers utilising ionised gases - identified in the review paper by Gudzenko et al[5]. Both are gas discharge lasers but, in the former, the active medium amplifies the radiation during ionisation, ie in the transition from a gas to a plasma. Examples of these collisionally-excited lasers include all CW and many pulsed systems (including the copper and gold vapour lasers). Most of the work performed on gas discharge lasers has been concentrated in this field. In plasma, or recombination, lasers

on the other hand, the amplifying medium is a rapidly recombining plasma. Ions created by means of an electrical discharge current recombine with electrons to form highly-excited atoms/ions. A population inversion $\bar{N}$ is established as these excited species relax to the ground state. Qualitatively, the two types differ in the direction of the deviation from thermodynamic equilibrium. In the former, the electron temperature T_e is higher than the equilibrium temperature T_i, whereas for recombination lasers, the plasma is supercooled ($T_e < T_i$). For pulsed electrical excitation, this means that the former use the leading edge of the discharge current pulse and the latter use the afterglow.

Collisionally-excited	$dI/dt > 0$	: $\bar{N}$ during discharge establishment and/or maintenance
Recombination	$dI/dt = 0$	: $\bar{N}$ only during discharge decay

Advantages of recombination as a method of creating a population inversion appear to be:

(i) The recombination flux between excited atomic/ionic states of a plasma has been used to obtain laser radiation on a broad range of discrete wavelengths - from the infra-red to the ultra-violet - in a wide range of chemical elements. In particular, the use of multiply-charged hydrogen-like ions to produce short wavelength coherent radiation - blue, ultraviolet and even soft X-rays - has been demonstrated[6].

(ii) Since collisional transitions predominate between the closely-spaced levels involved in the laser transition, radiation trapping becomes insignificant and the possibility exists for volumetric scaling of the active medium (cf the copper vapour laser where radiation trapping is a limiting factor).

(iii) In recombination lasers, the energy levels are populated "from above" as the excited species relax towards the ground state. This produces efficient pumping of the upper laser level while the stray pumping of the lower level is less pronounced.

1.3 HISTORY OF DEVELOPMENT - THEORETICAL

Recombination between electrons and atomic ions in an optically thin plasma is reviewed in the paper by D. R. Bates[7]. The use of the recombination regime as a possible pumping mechanism was first proposed by Gudzenko et al[8,9] who outlined the feasibility of achieving a population inversion in an optically-thin hydrogen-ion plasma. These (simple) numerical studies were applied to a pulsed recombination system consisting of a strongly-ionised, supercooled plasma which revealed that hydrogen was not the best medium for recombination. This was confirmed by further, more detailed analysis[10]. More promising were hydrogen-like ions with more complex electronic structures (eg lithium[11]) which, by including a simultaneous analysis of the behaviour of the electron temperature, suggested the possibility of achieving frequencies in the infra-red or visible spectra.

It is relevant at this point to consider both the plasma conditions and the energy level requirements of systems in order to assess their suitability as a recombination laser.

1.3.1 COLLISION KINETICS - EXPERIMENTAL REQUIREMENTS

The decay of an ionic species N_i upon removal of the excitation pulse is predominantly a three-body process, the rate of which is given by

$$\frac{dN_i}{dt} = - \alpha N_i n_e^2 \quad , \qquad (1.1)$$

where n_e is the free electron density and α is the three-body recombination coefficient. The latter is a function of ionic charge z and electron temperature T_e[1]

$$\alpha = 1.8 \times 10^{-8} z^3 L T_e^{-9/2} \quad , \qquad (1.2)$$

where $L = \log_e(z^2+1)^{1/2}$ is used in place of the more familiar Coulomb logarithm[12].

From (1.1), we get

$$N_i(t) = N_i(0) \exp(-t/\tau_r) \qquad (1.3a)$$

where the time constant characteristic of recombination is

$$\tau_r = \frac{1}{\alpha n_e^2} \quad . \qquad (1.3b)$$

Hence, the rate of recombination increases strongly with decreasing T_e and also with increasing n_e, N_i and z. Several methods of free electron cooling have been proposed, including diffusion to the walls[13], adiabatic expansion[14] and collisions with the particles of a light buffer gas at high pressure[13]. A high free electron density ensures that the probability of collisional transitions within groups exceeds the probability of optical transitions. Finally, the recombination rate exhibits a critical dependence on ionic charge z. Despite the increased energy input required, equation (1.2) indicates that

the rate for z=2 and z=3 ions are respectively 20 and 90 times greater than for singly-charged particles. Note, however, that from equation (1.3a), the duration of the recombination pulse rapidly decreases with increasing z.

1.3.2 ENERGY LEVEL REQUIREMENTS

During the non-equilibrium period of recombination which is characterised in the immediate afterglow by high values of n_e and low T_e, the rate of electron de-excitation from an upper level 2 to a lower level 1 is given by[15]

$$F_{21} = \langle \sigma_{21} v_e \rangle \alpha f_{12} \frac{g_1}{g_2} \frac{1}{E_{21} T_e^{1/2}} \quad . \tag{1.4}$$

The inverse process (electron excitation) is found from the principle of detailed balance to be

$$F_{12} = \langle \sigma_{12} v_e \rangle \alpha f_{12} \frac{1}{E_{21} T_e^{1/2}} \exp\left[\frac{-E_{21}}{kT_e}\right] \quad . \tag{1.5}$$

Here, E_{21} is the energy level separation, g_1 and g_2 are the respective statistical weights of the levels and f_{12} is the oscillator strength. From equations (1.4) and (1.5), both electron excitation and de-excitation rates increase with decreasing separation of energy levels. If T_e is sufficiently low, then the rate of electron de-excitation exceeds the excitation rate. This difference is further enhanced in ionic spectra.

Very often, the energy levels form distinct groups whose separation greatly exceeds that between levels within a group (Figure 1.2). Then, from equations (1.4) and (1.5), as n_e increases and T_e decreases, a Boltzmann distribution is established within each group. Hence, a population inversion may exist between

the individual lower levels of the upper group and the individual upper levels of the lower group, even though the total number density of the lower group exceeds that of the upper.

If the energy levels are widely spaced, the situation differs from that described above but the requirements of low T_e and high n_e still hold.

The atomic energy level and electrical requirements for efficient upper laser level pumping leading to recombining amplification are thus formulated as follows[16].

(1) The lower (upper) laser level should be one of the highest (lowest) in the lower (higher) group of closely spaced levels.

(2) Transitions between the upper and lower groups should be optically allowed.

(3) The electron density should be sufficiently high.

(4) The electron temperature should fall rapidly after removal of the excitation pulse.

(5) The transition should occur in doubly or triply ionised species.

(6) The ionisation potentials of the active medium and buffer gas atoms should be respectively as low and as high as possible.

The considerations outlined above suggest that the problem of operating an efficient recombination laser can be viewed as achieving two conditions:

(a) the establishment of a rapid de-population of the lower level of the laser transition

and (b) the generation of a rapidly recombining plasma.

1.4 LABORATORY DEVICES

The first demonstration of a recombination laser was by Latush and Sèm in 1972[17,18]. Lasing was observed in the afterglow of discharges containing the vapours of doubly ionised alkaline-earth metals. This group of metals - beryllium, magnesium, calcium, strontium, barium and radium - satisfies the energy level requirements outlined in Section 1.2. For example, the upper laser level of SrII (6S, Figure 1.2) is the lowest among the 4F, 5D and 6S group of levels (Group 2) and the lower laser level (5P) is the highest of the 5P, 4D and 5S group (Group 1). Strontium and calcium exhibited the greatest output powers at 430.5nm and 373.7nm respectively.

From these first devices of low output power, research on strontium lasers has been exclusively reported by two Soviet ([19,20] and [21]) and two Australian ([22-27] and [28]) groups. To date, strontium has exhibited the best performance (Bukshpun[20]), the maximum output on the 430.5nm line being 3.9W. This was achieved from a 33cm long, 1cm diameter water-cooled beryllia discharge tube at a pulse repetition frequency of 29kHz. This corresponds to a specific power of 150 $mWcm^{-3}$.

Zhukov et al[16] have demonstrated 0.5W on the 373.7nm line of calcium. Stimulated emission has also been achieved in beryllium [29], magnesium[17] and barium[17]. Note that, although the latter has been operated in the recombination regime, the self-terminating IR transition has produced the greatest output powers [30-32]. At present, the group IIa recombination lasers provide high peak power radiation at wavelengths from the UV (371 nm) to the IR (1248 nm). Table 1.1 lists the relevant wavelengths and transitions.

Element	Transition	λ(nm)	Buffer gas	Optimum temperature (^{o}C)	Reference
BeII	$4^2S_{1/2}$-$3^2P_{3/2}$	527.2	He	1300	30
MgII	$4^2P_{3/2}$-$4^2S_{1/2}$	921.8	He		17
	$4^2P_{1/2}$-$4^2S_{1/2}$	924.4	He		"
CaII	$5^2S_{1/2}$-$4^2P_{1/2}$	370.6	He	780	17,18,19
	$5^2S_{1/2}$-$4^2P_{3/2}$	373.7	He	"	"
	$6^2S_{1/2}$-$5^2P_{3/2}$	994.0	He,Ne	"	"
SrII	$6^2S_{1/2}$-$5^2P_{1/2}$	416.2	He,Ne+H_2	680	17,18,19
	$6^2S_{1/2}$-$5^2P_{3/2}$	430.5	He,Ne+H_2	"	"
	$6^2D_{5/2}$-$6^2P_{3/2}$	870.0	He,Ne	"	"
	$7^2S_{1/2}$-$6^2P_{1/2}$	1087.0	He,Ne	"	"
	$7^2S_{1/2}$-$6^2P_{3/2}$	1123.0	He,Ne	"	"
BaII	$5^2G_{7/2}$-$4^2F_{5/2}$	677.1	He,Ne+H_2		17
	$5^2G_{9/2}$-$4^2F_{7/2}$	687.6	He,Ne+H_2		"
	$9^2D_{5/2}$-$8^2P_{3/2}$	1109.2	He,Ne+H_2		"
	$8^2S_{1/2}$-$7^2P_{1/2}$	1158.1	He,Ne+H_2		"
	$10^2S_{1/2}$-$8^2P_{3/2}$	1193.5	He,Ne		"
	$8^2S_{1/2}$-$7^2P_{3/2}$	1247.8	He,Ne+H_2		"

Table 1.1. Group IIa metal vapour recombination laser transitions

All of the results described above were obtained using a longitudinal electrode configuration within the laser head. Recombination lasers have been operated with transverse structures ([33-36] for Sr^+ and [37] for Ca^+), albeit with lower output powers. However, the combination of high pressure and high peak current pulse obtainable from a transverse design have produced an order of magnitude greater extraction energies[34,37] and indicate that greatly increased average output powers may be obtainable.

Recombination lasers hold great promise as very high power sources of coherent radiation at wavelengths down to the VUV and XUV which have proved

inaccessible with other active media. For a constant gain amplification factor, the required fluorescent emission per unit volume on the laser transition scales as frequency $\upsilon^{4.5}$. The isoelectronic scaling of a rapidly recombining plasma to produce X-ray laser radiation was proposed by Molchanov[38]. This was followed up by intense activity at Lawrence Livermore National Laboratory in a series of theoretical and large-scale experimental studies using the two most powerful beams in the world from the neodymium glass laser Novette[39]. This extended the lasing spectrum to 20.6nm, the previous limit having been set at 116nm in 1972 with the Werner band H_2 laser. Further studies on carbon plasmas have resulted in observations of population inversions in the XUV region[40,41]. To date the shortest wavelength achieved is 4.55nm in MgXIII in a collaborative experiment between the Rutherford Appleton Laboratory, Oxon., and Osaka University, Japan[42].

The availability of MW and GW single pulses at wavelengths of 25nm to perhaps 2.5nm will revolutionise X-ray methods in a manner similar to that achieved by optical lasers. Applications include soft X-ray microscopy, material lithography on a submicron scale and the study of non-linear optical effects at XUV frequencies. However, it is expected that these lasers will only be available at the national laboratories where there exist the powerful optical frequency lasers to create the plasmas.

Despite such promise, the output power of devices built to date have been modest compared with gas lasers such as copper vapour (pulsed power outputs in excess of 200W[43]). The relative ease of achieving a population inversion in conventional gas lasers has led to the comparative neglect of the advantages to be gained from recombination and, even now, experimental work lags behind the theory. There may be two reasons for the relatively poor performance of

recombination lasers. Firstly, the short electronic relaxation times of dense plasmas impose stringent requirements on the population and depopulation of the upper and lower laser levels. The second reason is an experimental limitation and the one to which this thesis is addressed; the technology to generate large-volume rapidly-recombining plasmas of sufficient density and uniformity is not yet fully developed. The latter cause may be further sub-divided. In a self-heated strontium laser, an electrical discharge operating at high repetition rate heats the buffer gas to the lasing temperature. However, the temperature range for stimulated emission is very narrow (approximately $50^{o}C$, for the reasons explained in Chapter Four) and in order to produce a thermally homogeneous plasma over a "reasonable" volume, the radial temperature gradients must be reduced. At present, the maximum volumetric scaling is to tube bores of 13mm. Some attempts have been made to scale the output power of the SVL[23,26] and CaVL[44] by using tubes of larger cross-section but, although total pulse energies increased, average powers showed no marked improvement. Thermal relaxation processes can be accelerated by employing discharge tubes of rectangular cross-section due to the proximity of the walls. This is presently being investigated[25]. Although double-pulse experiments have indicated that pulse repetition frequencies up to and beyond 1 MHz are theoretically possible[22], this leads to excessive heating of the active volume which limits laser output power. The second limitation requires consideration of the lasing mechanism itself. A slowly-decaying current pulse will only serve to re-populate the lower-lying states, including the lower laser level, and destroy the population inversion. Thus, it is advantageous to produce a high amplitude current pulse (to achieve a large Sr^{++} density) with as rapid a termination as possible. In particular, the decay time of the discharge current pulse τ_p should be short compared with the characteristic recombination time of the plasma, ie

$$\tau_p << \tau_r \quad . \tag{1.6}$$

To date, insufficient attention has been paid to the pulsed power aspect of recombination lasers and yet it is this that determines the performance. What is needed is a parametric study of the driving circuit as a whole. Modelling of the modulator is hindered by the fact that the most common and powerful circuit simulators - SPICE and MICROCAP - are essentially only applicable to linear time-invariant circuits. Hence, they are rendered useless in this case. Furthermore, computer coding of the strontium laser kinetics is an essential but neglected topic.

1.5 OBJECTIVES

This research project is concerned with producing a strontium vapour laser of high average power, thereby realising the promise of recombination lasers outlined in the previous section. A compact, portable modulator/recombination laser unit could be used in laboratory, industry and health centre and hospital applications. It is proposed that high average power may be achieved by concentrating on the pulsed power circuitry in order to produce an extremely rapidly-terminated, well-matched discharge current pulse. To fully understand and improve the performance of the recombination laser circuit, it was decided to develop a completely general computer program for high peak power and repetition rated systems. To this end, GCAP (Generalised Circuit Analysis Program) was written and is described in Chapter Two. Based on the powerful technique of state variable analysis (SVA), GCAP represents a pulsed power equivalent of MICROCAP, retaining the full range of analysis options but being applicable also to non-linear and time-varying circuits. Furthermore, GCAP contains a library of commonly-used devices and options by which the user may

define complex circuit elements. Any system that can be modelled as a collection of linear or non-linear, time-invariant or time-varying circuit elements can be analysed. GCAP is written in standard PASCAL and may be run on mainframe or personal computers.

One of the most interesting aspects of SVA is that cases consisting of a wide range of physical systems which are interconnected but which operate under a different set of laws can be analysed simultaneously, ie the system can be viewed as one mathematical model. In Chapter Three, this principle is applied to analyse the pulsed gas laser system. The atomic (number density of particles in a particular state), optical (number density of photons), thermodynamic (local particle temperatures and pressures) and electrical (voltage and current) internal states of the system are the state variables. This provides a novel insight into the development of the plasma, for example the build-up and decay of the laser pulse.

One of the main advantages of GCAP is that the circuit equations can be combined with equations describing the behaviour of gas discharge parameters. In Chapter Four, the effects of systematic electrical circuit (capacitor charging voltage, storage capacitance and pulse repetition frequency) and thermodynamic (tube temperature and buffer gas pressure) parameter variation on species densities and consequently laser pulse energy and average output power are studied. Kinetic coding of the strontium vapour laser provides an indication of the requirements of the modulator circuit. Computer analysis reveals that for a narrow-bore (13mm i.d., to reduce radial temperature variations), high-pressure ($\geq$150 mbar, to increase volume recombination of excited strontium species by collisions with buffer gas atoms) strontium vapour laser, the optimum voltage applied to the laser head (longitudinal discharge

design) should be between 10 and 30kV, delivering in excess of 150A, corresponding to a current density of $\geq 100 Acm^{-2}$. The precise values of these parameters depend on the electrode separation and temperature, pressure and volume of the active medium together with the storage capacitance and pulse repetition frequency of the excitation circuit. The risetime of the current pulse is not as critical as in the copper vapour laser; a figure of the order of 100 to 200ns is acceptable. However, the analysis reveals that the stimulated emission would be greatly enhanced if the fall time were reduced by an order of magnitude with respect to this figure. This result is supported by Gudzenko et al[5] in which it is concluded that a population inversion may be obtained if the free electrons are cooled by a rapidly terminated ionising pulse in a time τ_{Te} which is considerably shorter than the electron density relaxation time τ_{ne}. Furthermore, the current pulse must not be allowed to oscillate as the repeated cycles will re-heat the plasma and destroy the population inversion. The current overshoot must be kept as low as possible. Table 1.2 summarises the modulator requirements.

Laser Cathode Voltage	:	10 - 30 kV
Peak Forward Laser Current	:	100 - 1400 A
Overshoot Current	:	Minimal
Pulse Repetition Frequency	:	100 Hz - 30 kHz
Rate of Rise of Current	:	10^9 As^{-1} (10% - 90%)
Rate of Fall of Current	:	$>10^{10}$ As^{-1} (90% - 10%) (corresponding to <75ns)

Table 1.2. Recombination laser modulator requirements

Chapter Five provides experimental validation of the computer simulations

based upon the modulator/laser system shown in Figure 1.3. A high voltage power supply resonantly charges a pulse forming network to twice the supply voltage via a charging (and a load bypass) element. A closing switch is triggered and discharges the storage capacitance through the laser load.

The power supply is a conventional transformer-isolated step-up design capable of supplying 20kV DC at half an amp. Chapter Five also reviews the high-power closing switches used to transfer energy to the discharge. The switch must be capable of switching kilovolts at high peak currents in the multi-kilohertz region. Metal vapour laser circuits almost invariably employ thyratrons due primarily to their high voltage hold-off capability and reliability[45,46]. However, the thyratron has limited peak current and di/dt ratings. The required pumping pulse amplitudes are typically of magnitude up to 1kA with a duration of a few tens to 500ns, holding off voltages up to 30kV at pulse repetition frequencies of 100Hz to 30kHz.

The operating characteristics and test results of a discharge-heated longitudinal strontium vapour laser are presented. An average power of approximately 0.5W is reliably obtained under sealed-off conditions. There is good agreement between model predictions and system performance over the range of experimental conditions encountered.

Recombination laser performance depends critically on the shape of the applied discharge current pulse. Chapter Six describes a technique of pulse shaping in which the bypass element is a saturable inductor. The magnetic switch is conventionally used to decrease the current pulse rise time. However, in the application described here, it is the fall time which is reduced. Saturation of the magnetic core provides an alternative path for the discharge

current and rapidly terminates that through the laser head. Furthermore, the point in time at which saturation occurs can be controlled by appropriate magnetic core biasing.

A detailed design example of the use of GCAP is included as an indication of the method by which the designer interacts with the program. GCAP is used to estimate the theoretical limits on the fall-time of the laser current by analysis of a flux-controlled model of the saturable inductor. The design parameters relevant to two methods of biasing the core (pulsed and DC) are outlined. This circuit generates current pulses with peak amplitudes up to 300A and fall times of less than 70ns (90%-10%).

REFERENCES FOR CHAPTER 1

1. I. S. Veselovskii, "Electron recombination coefficient with three-body collisions in a plasma", Sov. Phys. - Tech. Phys., 14, No.2, p.193 (1969).

2. C. E. Little, M. D. Ainsworth and J. A. Piper, "On the feasibility of a 'white-light' Au-Cu-Sr^+ laser", Oyo Buturi, 57, No.7, pp.1047-1052 (1988).

3. R. W. Waynant, "Medical applications of ultraviolet lasers", SPIE vol.894, *'Gas Laser Technology'*, Los Angeles, CA, pp.60-68 (1988).

4. M. Hamza and M. Hamza, "Laser transcutaneous bilirubin meter: a new device for bilirubin monitoring in neonatal jaundice", SPIE vol.907, *'Laser Surgery: Characterization and Therapeutics'*, Los Angeles, CA, pp.146-149 (1988).

5. L. I. Gudzenko, L. A. Shelepin and S. I. Yakovlenko, "Amplification in recombining plasmas (plasma lasers)", Sov. Phys. - Usp., 17, No.6, p.848 (1975).

6. M. H. Key, "XUV Lasers", J. Mod. Opt., 35, No.3, p.575 (1988).

7. D. R. Bates, A. E. Kingston and R. W. P. McWhirter, "Recombination between electrons and atomic ions. I. Optically thin plasmas", Proc. Roy. Soc., A267, p.297 (1962).

8. L. I. Gudzenko and L. A. Shelepin, "Radiative enhancement in a recombining plasma", Sov. Phys. - Dokl., 10, No.2, p.147 (1965).

9. L. I. Gudzenko, A. T. Mamachun and L. A. Shelepin, "Hydrogen level populations in a pulsed recombination plasma", Sov. Phys. - Tech. Phys., 12, No.5, p.598 (1967).

10. B. F. Gordiets, L. I. Gudzenko and L. A. Shelepin, "Relaxation processes and amplification of radiation in a dense plasma, Sov. Phys. - JETP, 28, No.3, p.489 (1969).

11. B. F. Gordiets, L. I. Gudzenko and L. A. Shelepin, Zh. Prikl. Mekh. Tekh. Fiz, No.6, p.115 (1968).

12. H. W. Drawin, "Non-equilibrium plasmas", in 'Reactions Under Plasma Conditions: Volume 1', ed. M. Venugopalan, Wiley-Interscience, New York, Ch. 3, Section IV, p.149 (1971).

13. B. F. Gordiets, L. I. Gudzenko and L. A. Shelepin, "Cooling of free electrons in a plasma", Sov. Phys. - Tech. Phys., 11, No.9, p. 1208 (1967).

14. L. I. Gudzenko, S. S. Filippov and L. A. Shelepin, "Rapid recombination of plasma jets", Sov. Phys. - JETP, 24, No.4, p.745 (1967).

15. B. E. Cherrington, "Distribution functions and the Boltzmann equation", in 'Gaseous Electronics and Gas Lasers', First Edition, Pergamon Press, Ch.4, pp.53-57 (1980).

16. V. V. Zhukov, E. L. Latush, V. S. Mikhalevskii and M. F. Sèm, "Recombination lasers utilizing vapours of chemical elements. I. Principles of achieving stimulated emission under recombination conditions", Sov. J. Quantum Electron., 7, No.6, p.704 (1977).

17. E. L. Latush and M. F. Sèm, "Stimulated emission due to transitions in alkaline-earth metal ions", Sov. J. Quantum Electron., 3, No.3, p.216 (1973).

18. E. L. Latush and M. F. Sèm, "Laser recombination transitions in CaII and SrII ions", Sov. Phys. - JETP, 37, No.6, p.1017 (1973).

19. V. V. Zhukov, V. S. Kucherov, E. L. Latush and M. F. Sèm, "Recombination lasers utilizing vapours of chemical elements. II. Laser action due to transitions in metal ions", Sov. J. Quantum Electron., 7, No.6, p.708 (1977).

20. L. M. Bukshpun, E. L. Latush and M. F. Sèm, "Influence of the temperature of the active medium on the stimulated emission characteristics of an Sr-He recombination laser", Sov. J. Quantum Electron., 18, No.9, p.1098 (1988).

21. V. E. Prokop'ev and V. I. Solomonov, "Investigation of a strontium vapour laser", Sov. J. Quantum Electron., 15, No.6, p.832 (1988).

22. M. S. Butler and J. A. Piper, "Optimization of excitation channels in the discharge-excited Sr^+ recombination laser", Appl. Phys. Lett., 45, No.7, p.707 (1984).

23. M. S. Butler and J. A. Piper, "Pulse energy scaling characteristics of longitudinally excited Sr^+ discharge recombination lasers", IEEE J. Quantum Electron., QE-21, No.10, p.1563 (1985).

24. C. E. Little and J. A. Piper, "Development of efficient high-power violet Sr^+ and ultraviolet Ca^+ recombination lasers", SPIE vol.894, *'Gas Laser Technology'*, Los Angeles, CA, pp.121-127 (1988).

25. C. E. Little and J. A. Piper, "High-power violet Sr^+ recombination lasers", SPIE vol.1041, *'Metal Vapour, Deep Blue, and Ultraviolet Lasers'*, Los Angeles, CA, pp.167-174 (1989).

26. C. E. Little and J. A. Piper, "Average-power scaling of self-heated Sr^+ afterglow-recombination lasers", IEEE J. Quantum Electron., QE-26, No.5, p.903 (1990).

27. J. A. Piper and C. E. Little, "Discharge-excited metal-atom plasma recombination lasers", Japan-Australia Workshop on Gaseous Electronics and its Applications, CSIRO Nat. Measurements Lab., Sydney, Jan 18-22, 1989.

28. C. W. McLucas and A. I. McIntosh, "Discharge heated longitudinal Sr^+ recombination laser", J. Phys. D, 19, p.1189 (1986).

29. V. V. Zhukov, V. L. Il'yusko, E. L. Latush and M. F. Sèm, "Pulse stimulated emission from beryllium vapour", Sov. J. Quantum Electron., 5, No.7, p.757 (1975).

30. B. G. Bricks, T. W. Karras and R. S. Anderson, "An investigation of a discharge-heated barium laser", J. Appl. Phys., 49, No.1, p.38 (1978).

31. V. V. Kazakov, S. V. Markova and G. G. Petrash, "Investigation of physical processes in a pulsed barium vapour laser", Sov. J. Quantum Electron., 14, No.5, p.642 (1984).

32. P. A. Bokhan, "Mechanism limiting the pulse repetition frequency of a barium vapour laser", Sov. J. Quantum Electron., 16, No.8, p.1041 (1986).

33. A. M. Bogus, V. L. Dzhikiya and A. A. Chernov, "Apparatus for investigating stimulated emission from transverse discharges in pure metal vapours", Sov. J. Quantum Electron., 8, No.2, p.259 (1978).

34. M. Brandt, "Transversely excited Sr^+ recombination laser", Appl. Phys. Lett., 42, No.2, p.127 (1983).

35. M. Brandt, "Repetitively pulsed transversely excited Sr^+ recombination laser", IEEE J. Quantum Electron., QE-20, No.9, p.1006 (1984).

36. M. S. Butler and J. A. Piper, "High-pressure, high-current transversely excited Sr^+ recombination laser", Appl. Phys. Lett., 42, No.12, p.1008 (1983).

37. M. S. Butler and J. A. Piper, "High-power transverse-discharge Ca^+ recombination laser", Appl. Phys. Lett., 43, No.9, p.823 (1983).

38. A. G. Molchanov, "Lasers in the vacuum ultraviolet and in the X-ray regions of the spectrum", Sov. Phys. - Usp., 15, pp.124-129 (1972).

39. S. Slutz, G. Zimmerman, W. Lokke, G. Chapline and L. Wood, "Concerning the creation and utilisation of population inversions in spatially anisotropic, dense, high Z plasmas", Bull. Am. Phys. Soc., 17, p.972 (1972).

40. M. H. Key, C. L. S. Lewis and M. J. Lamb, "Transient population inversion at 18.2nm in a laser produced CVI plasma", Optics Commun., 28, No.3, pp.331-335 (1979).

41. D. Jacoby, G. J. Pert, S. A. Ramsden, L. D. Shorrock and G. J. Tallents, "Observation of gain in a possible extreme ultraviolet lasing system", Optics Commun., 37, No.3, pp.193-196 (1981).

42. H. Daido, P. R. Herman, T. Jitsuno, Y. Kato, E. Miura, S. Nakai, H. Nishimura, H. Shiraga, T. Tachi, H. Takabe, M. Takagi, C. Yamanaka, M. Yamanaka, G. J. Pert, M. H. Key, P. T. Rumsby, S. J. Rose and G. J. Tallents, "ILE/UK joint experiment on X-ray lasers", Rutherford Appleton Laboratory report no. RAL-88-042-A1-2 (1988).

43. N. Aoki, H. Kimura, C. Konagai, S. Shirayama, T. Miyazawa and T. Takahashi, "High-power copper vapor laser development", SPIE vol.1412, *'Gas and Metal Vapor Lasers and Applications'*, Los Angeles, CA, pp.2-11 (1991).

44. C. E. Little and J. A. Piper, "Average-power limitations of large-aperture self-heated Ca^+ afterglow-recombination lasers", Optics Commun., 68, No.4, p.282 (1988).

45. H. Menown and C. V. Neale, "Thyratrons for short pulse laser circuits", IEEE Thirteenth Pulse Power Modulator Symposium, Buffalo, NY, pp.125-128, June 20-22, 1976.

46. P. D. Culling, H. Menown, C. A. Pirrie and C. A. Roberts, "Recent developments in high repetition rate thyratrons for copper vapour lasers", Sixth IEEE Pulsed Power Conference, Arlington, VA, pp.598-603, June 29-July 1, 1987.

FIGURES FOR CHAPTER 1

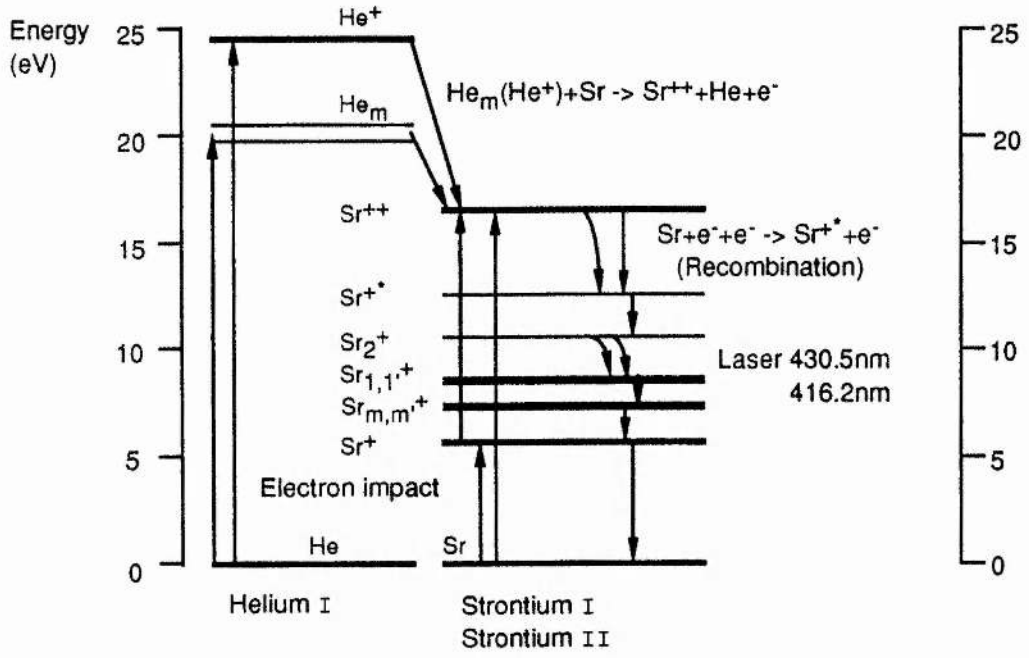


Figure 1.1. Energy level diagram for the helium-strontium laser

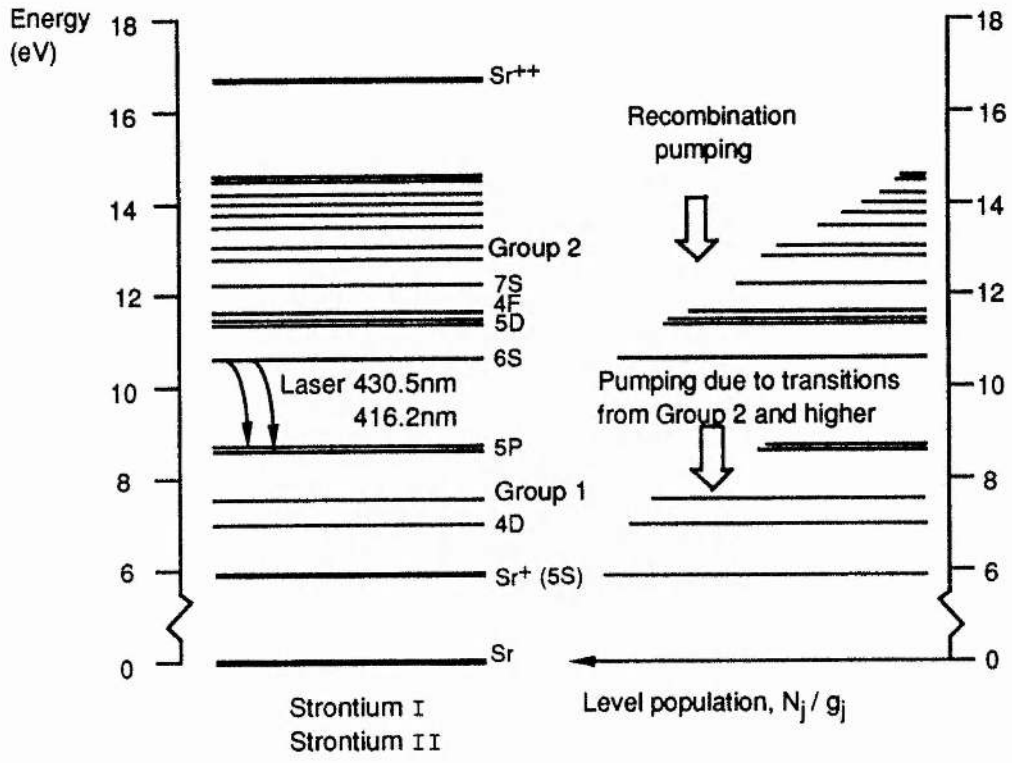


Figure 1.2. Energy level and term diagram of SrII illustrating Boltzmann distributions within the two groups of levels and a population inversion between these groups

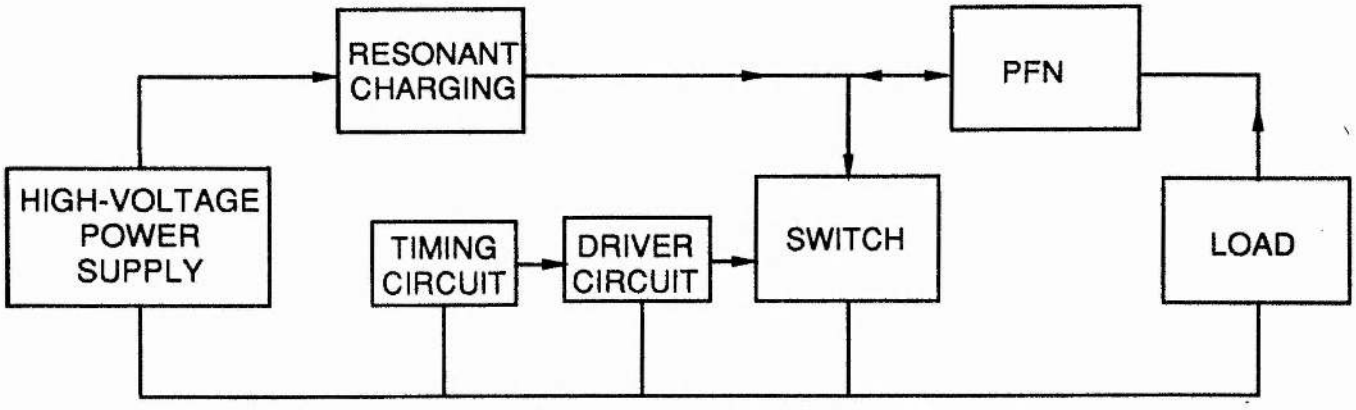


Figure 1.3. Line type modulator block diagram

Chapter 2

G C A P : A Generalised Circuit Analysis Program for Pulsed Power

2.1 MOTIVATION

In the study of high power pulse generation, system modelling plays an important role both in the design of systems and in producing a quantitative understanding of their performance. Networks are usually designed assuming a constant resistive load. However, power modulators contain variable circuit elements and often supply loads with time-dependent impedance. High peak power gas discharge devices have created the need for pulse generators operating into non-linear and time-varying reactive loads with specifications of pulse duration, magnitude, repetition rate, etc. Predictions of system performance by approximate circuit simulation software such as SPICE[1] or MICROCAP[2] suffer from the inability of these programs to model the precise behaviour of continuously changing circuit components such as saturable magnetics, switches and gas discharge loads. Several papers have been devoted to this subject. In some cases[3,4], the techniques are based on modifying existing codes to simulate actual conditions. More satisfactory are efforts to develop programs applicable to specific systems[5-10] but it is precisely this lack of generality which limits interest. Little work appears to have been published on the problem of predicting system performance with loads of time-varying or non-linear impedance.

The circuit simulation code developed aims to increase understanding of the behaviour of thyratron-switched modulators used to power metal vapour lasers

and, in particular, to study the interaction of power supply, modulator and the highly non-linear gas discharge loads.

The requirements of the program are as follows.

(1) It should serve as a fast, interactive tool for the study of non-linear, time-varying circuits.

(2) The system should be user-friendly in a manner similar to the commercial models SPICE and MICROCAP.

The resulting program, known as GCAP (Generalised Circuit Analysis Program) satisfies the above requirements. The basis of GCAP is the powerful method of state-variable analysis (SVA). The equations obtained are numerically integrated and the state matrices continuously updated to provide data related to circuit behaviour. GCAP has the ability to accomodate any time-varying and/or non-linear circuit element if it is defined either from a library inherent in the program or by user-specification. This opens up the field of pulsed power to analysis by computer.

This chapter describes the program structure. In Section 2.2, a brief background to the theory of SVA relevant to the program is presented. The approach to formulating a completely general program is then discussed. Finally, GCAP is discussed in terms of acceptable circuit components, the analysis options available and formatting of input and output data. The structure of the program is illustrated by means of flow diagrams.

2.2 THE METHOD OF STATE VARIABLE ANALYSIS

2.2.1 INTRODUCTION

The field of control system design has come to be dominated by the state-variable approach, and many important results have emerged from its application[11]. The method of SVA is based on the concept of the A matrix and has its foundations in Bashkow's A-matrix paper[12], and in further refinements by Bryant[13]. It considers the internal "state" of a network rather than the external behaviour described by other techniques. In addition to the improved insight into circuit operation that this provides, the method offers the following advantages.

(1) Describing the circuit in terms of SVA results in a set of first order differential equations whose numerical solution is ideally suited to computers. Furthermore, iterative solution of the equations enables both time-varying and non-linear networks to be studied by a simple modification to the A matrix.

(2) For the design of switched-mode power supplies, the technique of "state-space averaging" developed by Middlebrook[14] is a direct extension of the state-variable method.

(3) Matrix inversion yields results in the frequency domain which are valuable in discussing aspects of control system design such as bandwidth, gain and phase margins, and corner frequencies.

2.2.2 THEORETICAL BACKGROUND

Classical analyses of second-order circuits consisting of linear time-invariant elements are based on either nodal or mesh methods and lead in

general to a set of coupled second order scalar differential equations. However, the solutions become involved and require either detailed numerical analysis or Laplace transformation - neither of which is performed efficiently by a computer. Furthermore, this approach, which is used by the circuit simulators MICROCAP and SPICE, does have inherent limitations with regards to both non-linear and time-varying systems which restrict its range of applications, particularly in the field of pulsed power.

The technique of SVA removes the need for transformation to the frequency domain for solution and instead places emphasis on the time-domain behaviour, thereby involving a more direct intuitive approach. It relies upon the fact that any scalar differential equation of order n can be expressed as a minimal set of n first order differential equations. In canonical, or state-space, form these equations are

$$\dot{x}_1(t) = a_{11}x_1(t) + a_{12}x_2(t) + \ldots + a_{1n}x_n(t) + b_{11}u_1(t) + b_{12}u_2(t) + \ldots + b_{1m}u_m(t)$$
$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots$$
$$\dot{x}_n(t) = a_{n1}x_1(t) + a_{n2}x_2(t) + \ldots + a_{nn}x_n(t) + b_{n1}u_1(t) + b_{n2}u_2(t) + \ldots + b_{nm}u_m(t)$$

or, in vector form,

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \qquad \textbf{STATE EQUATION} \qquad (2.1)$$

and

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) + \mathbf{E}\dot{\mathbf{u}}(t) \qquad \textbf{INPUT-STATE-OUTPUT EQUATION} \qquad (2.2)$$

In these equations, $\mathbf{x}(t)$, $\mathbf{u}(t)$ and $\mathbf{y}(t)$ are, respectively, the state, input

and output vectors ($\mathbf{x} = (x_1, x_2, ..., x_n)^T$ is a real variable) and **A** to **E** are matrices. The formal definition of the "state" of the system is "the minimal amount of information necessary at any time to characterise completely any future behaviour of the system". The state variables $x_1, ..., x_n$ are the independent initial conditions which the system can support. The state model (equations (2.1) and (2.2)) completely determines the time-evolution of the system.

All properly modelled circuits have well-defined state equations. For the completely general case of a circuit containing both non-linear and time-varying elements, the system of n standard form state equations is given by

$$\begin{aligned} \dot{x}_1(t) &= f_1(x_1, x_2, ..., x_n, u_1, u_2, ..., u_n, t) \\ &\vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ \dot{x}_n(t) &= f_n(x_1, x_2, ..., x_n, u_1, u_2, ..., u_n, t) \end{aligned}$$

or,

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \qquad \textbf{NON-LINEAR STATE EQUATION} \qquad (2.3a)$$

Similarly, the output equation is

$$\mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), t) \qquad \textbf{NON-LINEAR INPUT-STATE-OUTPUT EQUATION} \qquad (2.3b)$$

In these equations, f and g are, in general, non-linear functions with time-varying elements. Since it is generally impossible to solve a non-linear differential equation (of any order) in closed form, the solution is found by

numerical iteration using a digital computer. A subroutine based on the Newton-Raphson algorithm[15] is commonly used.

For a lumped network consisting of linear time-invariant elements, independent and controlled sources as well as active and passive coupling elements, it is suitable to choose the state vector as the set of all capacitance voltages and all inductance currents

$$x(t) = \begin{bmatrix} V_c(t) \\ i_L(t) \end{bmatrix} . \qquad (2.4)$$

However, the voltage versus charge and current versus flux characteristics of time-varying elements are not constant. In order to emphasise the unified approach to constant and non-linear/time-varying elements, by choosing capacitor charges q_c and inductor fluxes ϕ_L defined by

$$q_c(t) = C(t)\, V_c(t) \qquad (2.5a)$$

$$\phi_L(t) = L(t)\, i_L(t) \qquad (2.5b)$$

where C(t) and L(t) are, in general, the small-signal capacitance and inductance at the operating points V_c and i_L, and are prescribed functions of time, then the new state vector

$$x(t) = \begin{bmatrix} q_c(t) \\ \phi_L(t) \end{bmatrix} \qquad (2.6)$$

applies universally. Since we have

$$q_c(t) = \int_{-\infty}^{t} i(\tau)d\tau \qquad (2.7a)$$

and

$$\phi_L(t) = \int_{-\infty}^{t} V(\tau)d\tau \qquad (2.7b)$$

then combining equations (2.5) and (2.7) we get

$$\dot{i}(t) = C(t)\dot{V}_c(t) + \dot{C}(t)V_c(t) \qquad (2.8a)$$

$$\dot{V}(t) = L(t)\dot{i}_L(t) + \dot{L}(t)i_L(t) \qquad (2.8b)$$

2.2.3 STATE EQUATION FORMULATION

Formulating the system of equations (2.1) and (2.2) is a topological problem. For simple networks (<10 components), the state equations can be derived by inspection. However, for complete generality and extension to more complex systems, the process is algebraically very involved and beyond the scope of this brief introduction. A full treatment is presented in the tutorial paper by Kuh and Rohrer[16] and in the book by Rohrer[17].

In summary, the network branches must be classified into a proper tree containing all of the independent voltage sources and capacitances, as many resistances as necessary (to include all nodes), but no inductances. The cotree will contain all independent current sources and inductances, the remaining resistances, but no capacitances. This information is used to formulate the fundamental cutset matrix Q, which in turn generates the fundamental submatrix F. The latter contains all the information concerning interconnections of branches and produces the set of fundamental circuit (f-circuit) equations. The remainder of the derivation of the state equations relies on matrix manipulation. The submatrix F generates further submatrices (expressing the

topological relationship between links and tree branches) which together with the element matrices (containing component values and characteristics) are used to derive firstly cutset and hence the hybrid *H* matrices. Finally, completely general explicit expressions are obtained for the state matrices **A** to **E** which can be substituted into the standard-form differential state and linear input-state-output equations (2.1) and (2.2).

2.2.4 TIME-DOMAIN ANALYSIS

The solution of equation (2.1), given an initial state $\mathbf{x}(t_o)$, is

$$\mathbf{x}(t) = \phi(t,t_o)\mathbf{x}(t_o) + \int_{t_o}^{t} \phi(t,\tau)\mathbf{B}(\tau)\mathbf{u}(\tau)d\tau \qquad (2.9)$$

where $\phi(t,t_o)$ is the state transition matrix, defined such that

$$\frac{d}{dt}\phi(t,t_o) = \mathbf{A}(t)\,\phi(t,t_o) \qquad (2.10a)$$

$$\phi(t,t) = \mathbf{I} \qquad . \qquad (2.10b)$$

For the linear time-invariant case, the solution of equation (2.1), $t_o = 0$, is

$$\mathbf{x}(t) = \exp(\mathbf{A}t)\mathbf{x}(t_o) + \int_{o}^{t} \exp(\mathbf{A}t)\mathbf{B}(\tau)\mathbf{u}(\tau)d\tau \qquad . \qquad (2.11)$$

This is the complete state response and it consists of two terms. First, the zero-input (state) response (ZIR)

$$\mathbf{x}_{zi}(t) = \exp(\mathbf{A}t)\mathbf{x}(t_o) \qquad (2.12)$$

is obtained with $\mathbf{u}(t) = 0$, ie this is the solution of $\dot{\mathbf{x}} = \mathbf{Ax}$. This provides complete information on the temporal behaviour of the network in terms of the energy storage elements as well as the desired outputs. Classically, this corresponds to the natural response. Most of the qualitative features of ZIR systems are preserved in non-linear circuits.

The second term in equation (2.11) is the zero-state (state) response (ZSR)

$$\mathbf{x}_{zs}(t) = \int_{t}^{t_o} \exp(At)\mathbf{B}(\tau)\mathbf{u}(\tau)d\tau \qquad (2.13)$$

corresponding to the response of the system to an excitation $\mathbf{u}(t)$ for a zero initial state ($\mathbf{x}(t_o) = 0$). Classically, this corresponds to the forced response.

2.2.5 TIME-DOMAIN SOLUTION OF THE STATE EQUATIONS

Numerical solution of the differential state equations (2.1) in the time-domain is obtained by means of the Runge-Kutta algorithm[18] which applies equally well to non-linear, time-variable differential equations. The algorithm is presented in Appendix A. It is a detailed and accurate method but yields estimates of calculation errors only with some effort[18]. The algebraic solution of the linear state equations (2.2) is straightforward. This iterative approach to solving the state equations provides SVA with one of its most powerful features. By updating the state matrices at each successive time step, non-linear and time-varying characteristics can be accounted for. However, the basis for the update must be known in the form of an equation describing the non-linear or temporal behaviour.

2.2.6 FREQUENCY-DOMAIN ANALYSIS

Solution of the state equations by the conventional method of Laplace transformation to the frequency-domain represents an inefficient approach. For example, an iterative algorithm based on the Bairstow-Hitchcock method[19] depends on an ability to factorise polynomials in the complex frequency variable, s. However, conversion to the frequency-domain is useful in that it provides important results - frequency-domain concepts such as "bandwidth", "gain and phase margins" and "corner frequencies" are entrenched in control system design[11]. Time-domain state-space methods serve to complement rather than displace the frequency-domain approach.

Laplace transformation of equations (2.1) and (2.2) yields

$$\mathbf{sX(s)} - \mathbf{x(0)} = \mathbf{AX(s)} + \mathbf{BU(s)} \qquad (2.14a)$$

$$\mathbf{Y(s)} = \mathbf{CX(s)} + \mathbf{DU(s)} \qquad (2.14b)$$

Hence, by combining equations (2.14a) and (2.14b), the frequency-domain output can be written

$$\mathbf{Y(s)} = \left[\mathbf{C(sI - A)^{-1} + D}\right]\mathbf{U(s)} + \mathbf{C(sI - A)^{-1}x(0)} \qquad (2.15)$$

where the first term on the right-hand side is the (transform) zero-state output, while the second term is the steady-state output.

The classical definition of the transfer function matrix **H(s)** relates the zero-state response $\mathbf{Y_{ZS}}$ to the excitation **U(s)**

$$\mathbf{Y(s)} = \mathbf{H(s)U(s)} \quad . \qquad (2.16)$$

Therefore, by inspection of equations (2.15) and (2.16), we get

$$\mathbf{H(s)} = \mathbf{C(sI - A)^{-1}B + D} = \mathbf{C}\,\frac{\mathbf{adj(sI-A)}}{\mathbf{det(sI-A)}}\,\mathbf{B + D} \quad . \qquad (2.17)$$

The poles of a system defined by **H(s)** are therefore eigenvalues of A[20], ie roots of the characteristic equation

$$\det(\mathbf{sI-A}) = 0 \qquad (2.18)$$

and the zeros (natural frequencies) of the system are the roots of[20]

$$\mathrm{adj}(\mathbf{sI-A}) = 0 \quad . \qquad (2.19)$$

Setting s = 0 in equation (2.17) yields the DC transfer function:

$$\mathbf{H(s)} = \mathbf{- CA^{-1}B + D} \quad . \qquad (2.20)$$

An important technique inherent in SVA is that of "shifting" poles by means of state feedback[20]. Consider Figure 2.1. The state equation of the open-loop system is given by equation (2.1) and its poles are the eigenvalues of **A**.

The closed-loop system has the state equation:-

$$\dot{\mathbf{x}} = \mathbf{Ax + B(u' - Kx)}$$
$$= \mathbf{(A - BK)x + Bu'} \qquad (2.21)$$

where **u'** is the input vector and **K** is a constant state feedback matrix. The poles are now described by the eigenvalues of **(A - BK)** and **K** can be chosen so that these eigenvalues have any desired real or imaginary values.

2.3 PROGRAM FORMULATION

2.3.1 AIMS OF PROGRAM

The overall design objectives that guide the development of a general circuit analysis program are discussed below.

A general simulation code should be capable of performing the following analyses:

(1) steady-state non-linear DC,

(2) steady-state small-signal AC

and (3) transient non-linear.

Additional capabilities built into the code might include the following.

(1) The program should have a convenient and simple input language to describe the circuit topology and element types, values and behavioural information. This language should also describe the analysis options required, the format of the output data and the time or frequency range over which this output is to be printed or plotted. No special knowledge of the computational technique should be required by the user to enter this information.

(2) The program should contain built-in models of the most commonly-encountered circuit elements such as MOSFETs, diodes, etc.

Furthermore, the user must be able to define new models to the program.

(3) For complete generality, the program must be able to accomodate non-linear and time-varying elements such as transistors, switches, saturable magnetics, gas discharge devices, etc. The models describing these elements will be either graphic, numerical, piece-wise linear (stored in look-up tables) or analytic.

(4) In addition to the DC, AC and transient analyses listed above, there should be a wide range of output options available. These might include transfer functions, poles and zeros, magnitude and phase response, etc. This output data will be in either tabular or graphics form.

(5) The designer should be able to interact with the program such that on obtaining the results of a simulation, the circuit may be modified until the desired results are obtained. Alternatively, the user may wish to study the sensitivity (or tolerance) of each element by parameter modification - these "worst-case" studies remove the need for the breadboard stage in a design. Thus, it is essential that the program has a fast execution time in order to place the designer in the "design feedback loop".

(6) Error checks are necessary in order to assess the reliability of the output.

(7) Finally, the program should have an open-ended, highly-modularised structure to allow the user to modify or extend the capabilities.

2.3.2 APPROACH

This section describes the approach to the development of a general circuit analysis program. A schematic flow diagram of the main program is shown in Figure 2.2. GCAP is written in Turbo Pascal Version 5.5. The subprogram capability allows a modular approach to the problem, where each module can be

refined in the initial development or extended by the user as necessary. In particular, the user-defined function capability permits the inclusion of models of circuit components.

Topological data describing circuit connections are first converted to a form acceptable to the program. If the user is only required to input a component identification name, type and the two (or more) nodal connections, then the program must formulate the fundamental cutset matrix Q from this data alone. An algorithm identifies the maximum and minimum node numbers, and calculates the total numbers of nodes, tree branches and links. A file (COMP1) is compiled consisting of component descriptors (see Section 2.4.2) arranged in ascending order of nodal connection number. An algorithm consisting of ten subprograms is included to classify the fixed topology network branches into a proper tree and generate the f-circuit equations internally. This is achieved by the following method:

(1) Scan COMP1 for an independent voltage source.

(2) Eliminate link voltage source if node numbers are duplicated.

(3) Store accepted component descriptors in array 'BRANCHES' in ascending order of nodal connection number.

(4) Eliminate duplicated node numbers.

(5) If all nodes are covered and connected then go to (7).

(6) Repeat steps (1) to (5) for capacitances, conductances and reciprocal inductances.

(7) Store remaining descriptors in array 'LINKS'.

Although this is a complicated procedure, it is only required to be performed at the beginning of a simulation.

The element matrices must be formulated taking account, if necessary, of any non-linearities in the circuit. For this reason, a DC analysis is run to determine the DC operating point of the circuit and the values of these components are calculated under these conditions. By combining these matrices with the fundamental sub-matrix F, derivation of the state matrices is straightforward. Each of these matrices is merely an array of numbers. Throughout this derivation, advantage is taken of the adjustable dimensions of matrices permitted within a subprogram. Once explicit forms of the state matrices **A** to **E** in equations (2.1) and (2.2) have been obtained, the user has the option of terminating the simulation with the DC analysis or, more usually, running either an AC, transient or frequency analysis.

A DC analysis is performed automatically prior to a non-linear transient analysis if it is necessary to determine the DC operating point for the initial conditions, and prior to an AC small-signal analysis to determine the small-signal linearised equivalent models of any non-linear devices. A Newton-Raphson algorithm[15] is included to find the DC operating point and is described in Appendix A. An initial guess at the element value under circuit conditions is taken to be half of the usual value. The Newton-Raphson formula is applied and a new value calculated. If a convergence is not achieved within ten steps, then the algorithm terminates and is repeated with a different initial estimate. This is the only analysis that requires changes to the circuit topology as all inductors are replaced by 1mΩ resistors and all time-varying sources, switches and capacitors are removed. This requires that the fundamental cutset matrix must be recalculated. A DC transfer curve is produced by applying a low-voltage excitation to the input node and monitoring the output at any selected point. An AC small-signal analysis operates on a

fixed set of state matrices describing the linearised circuit resulting from a DC analysis to calculate the response to a fixed magnitude, variable frequency sinewave.

Transient analysis is, in effect, a time domain simulation. Unlike AC analysis, where it is necessary to solve a fixed set of state equations, transient analysis requires that a new set of state equations be dynamically generated at each successive time-step, ie once the circuit has been simulated over one time increment, the state matrices are updated such that $\mathbf{x}(t)$, $\mathbf{y}(t)$ and $\mathbf{u}(t)$ are continuous; the final state in one configuration is the initial state in the subsequent configuration. This demonstrates an important advantage of the state-variable approach, namely its ability to analyse non-linear/time-varying circuits with no additional difficulty. As in the linear, time-invariant case, the state matrices are again derived from a fixed topology. However, in addition to updating these matrices due to the time increment, they are also updated as necessary due to the present state of individual components.

The state equations are solved numerically by means of a Runge-Kutta algorithm and the total, zero-input and zero-state response calculated. The time-step of calculation must be smaller than the smallest characteristic time-constant of the circuit. For this reason, if not specified, GCAP will automatically select the calculation increment by considering the eigenvalues of the dynamical matrix **A**. The step-value is then one-fifth of the smallest time constant explicit in the **A** matrix. The extremely stiff differential equations (widely spread time constants) pose a problem. The program continuously monitors the system and if all states remain constant for a time much greater than the calculation increment, then this parameter is increased to reduce

computer time. SPICE, on the other hand, formulates the more conventional second order differential equations and iteration is performed until an acceptable convergence is achieved for the transient solution. However, sometimes the program fails to converge and terminates, eg positive feedback circuits have proved to be a problem[1]. Furthermore, a displayed solution may be wildly incorrect.

Finally, a frequency analysis again operates on a fixed set of state equations. As indicated in Section 2.2.6, conversion to the frequency domain is not essential for solution, but is of immense interest. System poles, zeros and natural frequencies can be calculated together with stability checks and noise studies. Further, the frequency domain is a very useful area for studying feedback systems. GCAP has procedures to calculate closed-loop feedback system response as well as to tailor poles by state feedback.

In all cases, the simulation is performed over a user-specified time-range (for a transient analysis) or frequency-range (for AC and frequency analyses). Unless otherwise specified, the output of data is in tabular form. Alternatively, a graphics subprogram is included to plot these results, again over a user-specified range. Although the time increment and interval of calculation are previously set, the user has control over which data are to be printed or plotted at specified time or frequency points. This avoids the need for GCAP to repeat the entire simulation every time a different parameter is to be displayed.

2.4 GENERALISED CIRCUIT ANALYSIS PROGRAM

2.4.1 INTRODUCTION

GCAP is a general-purpose circuit simulation program. It is written in Turbo Pascal Version 5.5 and may be run on mainframe or IBM-compatible personal computers. GCAP is available on both 5.25" and 3.5" disks. The present version of GCAP contains approximately 10,000 executable statements and occupies about 2 MBytes of memory. Although this exceeds the limit of 640 KBytes of RAM inherent in MS-DOS, the use of "overlays" facilitates the development of large development projects. By this method, subprograms are loaded into memory when needed, and unloaded when not. Subprograms are used extensively (over one hundred in total) to perform a wide range of analysis, formatting and modelling functions. The program can cope with almost any component including those which may be non-linear, time-varying, and voltage- or current-sensitive. Any system that can be modelled as a collection of linear or non-linear, time-invariant or time-varying circuit elements (resistors, capacitors, inductors, independent or dependent (voltage- or current-controlled) voltage and current sources, etc.) can be analysed. GCAP has built-in models for the most common semiconductors and the user needs only to specify the relevant parameter values. These devices are the diode, BJT and MOSFET. With regard to the applicability of GCAP to the field of pulsed power, the user may define subcircuits in order to model devices. Thyratron switches, spark gaps and laser loads are examples of elements modelled by GCAP that are beyond the scope of MICROCAP. GCAP can produce four main categories of solution: non-linear DC, linear AC, non-linear transient and frequency analyses. Output data is in either tabular or graphics form.

This introduction is not meant to represent a complete guide to the use of GCAP - this information can be found in the user manual[21]. Rather, it is intended to give a brief description of the use of the program. The user interface is simple and straightforward. Interaction between the designer and GCAP is by means of three command types, namely component, analysis and structure descriptors. A descriptor is simply an input statement, the exact format of which is described in detail in Sections 2.4.2, 2.4.3 and 2.4.4. Descriptors are entered in the form of input files in any order and at any stage when running the program. For convenience, all the component and the majority of the analysis descriptors will be entered prior to program execution, whereas structure descriptors are individual statements which tend to be input at convenient stages.

2.4.2 ACCEPTABLE CIRCUIT COMPONENTS

The circuit to be analysed must first be defined to GCAP. The user need only specify element type and value, nodal connections and any relevant non-linear/time-varying data to completely describe the circuit. Component descriptors are used for this purpose and data entered as an input file (COMPON) at the beginning of the program, but they may be appended at any stage during the execution. Data is input on two separate lines, the first containing character data (to identify the element and its type) and the second numerical data (element value, nodal connections and any non-linear/time-varying parameters). To maintain generality, the user requires no special knowledge of the computational technique to enter this information. Numbers to designate nodes must be positive integers but need not be consecutive (see Section 2.3.2). An algorithm is used to arrange this information into a form acceptable to the program.

Examples of circuit components which come within the scope of GCAP are listed in Table 2.1.

Passive components:	Resistors, capacitors, inductors, independent and controlled voltage and current sources	
	Element	Model
GCAP library:	Transmission lines	N pi-networks consisting of series resistance and inductance and shunt capacitance and conductance
	Diodes	Square-law I-V characteristic
	BJTs	Ebers-Moll[22]
	MOSFETs	Schichman-Hodges[23]
User-defined examples:	Switches, time-varying sources, flux-controlled magnetic components, voltage-controlled gas discharge devices	

Table 2.1. Acceptable components

Modelling of Components

Library models are sub-circuits which, in GCAP, are considered as single elements. The library contains models based on well-documented models of predominantly semiconductor devices[22,23], the model relevant to each component being indicated in Table 2.1. The user needs only to specify the relevant model parameter values to define the element. User-defined models are also entered as sub-circuits. This feature allows simulation of components such as switches and saturable magnetic devices.

The time-dependent behaviour of components is entered explicitly in the form of equation (2.8). For the purpose of GCAP, opening and closing times of switches are divided into chosen intervals. New state matrices are formulated

at each successive time step. For continuity, the final switch states at any one time are the initial states at the next time step.

Non-linear dependence may be described either analytically or in piece-wise linear form. Examples of the former are the thyratron (Chapter Three) and the saturable bypass inductor (Chapter Six). The parameters used to describe non-linear elements are listed in Table 2.2. One of the two distinct state variables must be chosen for each element.

Element	Control parameter (State variable)
Capacitor	Voltage, V_c
	Charge, q_c
Magnetics	Current, i_L
(inductors, transformers)	Flux, ϕ_L
Switches	Voltage, V
	Current, i

Table 2.2. Control parameters for non-linear components

As an example, the general form of a component descriptor is

R R******* (LIN/GRA/POL/USE) (VAR/INV)

Value NA NB (Limit1) (Limit2) (Limit3) (Limit4) (Limit5)
(Limit6) (Limit7)

The first character, R defines the element to be a resistor and the second term is used to identify that element. 'Value' is the resistance in ohms and NA and NB are the nodal connections. This information is essential input to GCAP. Items in parentheses are optional. The first two are used to specify any

relevant non-linear/time-varying characteristics. The option LIN/GRA/POL/USE indicates whether the resistor is linear or non-linear (GCAP-modelled graphically, polynomially or by the user), respectively. The default value is LIN. VAR/INV specifies whether the resistor is time-varying or time-invariant, the default value being INV. Finally, the limits 1-7 are required to specify the relevant parameters if the options GRA/POL/USE (Limits 1-5) or VAR (Limits 6 and 7) are used. An example of a component descriptor is shown below:

```
R       RBYPASS     GRA
2.2E3   4   17      12E6   14E7   3
```

2.4.3 ANALYSIS OPTIONS

Analysis options may be expressed in terms of four categories and specified to GCAP at any stage during the program execution. The categories are DC, which generally aims to calculate the DC operating point of the circuit; AC, which examines the circuit response to sinusoidal input; the so-called transient category, in which temporal behaviour of the circuit is calculated in the time-domain, naturally; and the final category is "frequency", which is used for frequency domain calculations.

All analysis options are specified to GCAP in the form of an input file (ANALYS) at any stage during the program execution. Data is again input on two separate lines, the first containing character data (to identify the analysis to be performed) and the second numerical data (time or frequency range over which the simulation is to be run).

Descriptor	Analysis
Non-linear DC:	
DCTRAN	Calculates DC steady-state transfer function (corresponding to the AC transfer function with s = 0)
Linear AC:	
ACTRAN	Calculates AC transfer function
BODE	Produces a Bode plot based on the AC transfer function
GAINM	Calculates the gain margin of the circuit
PHASEM	Calculates the phase margin of the circuit
STABM	Calculates the stability margin of the circuit
Non-linear transient:	
TIME	Sets time increment and interval
TEMPRE	Sets the temperature of the transient analysis
IMPULS	Calculates the response to an impulse excitation
RESPON	Specifies type of response of interest (steady-state, transient or total)
SENSE	Performs a sensitivity check on each specified output variable with respect to every circuit parameter
Frequency domain:	
STABLE	Performs a stability check
POLES	Calculates poles of system
ZEROS	Calculates zeros of system
POLESH	Facility to shift poles of system
FBRESP	Calculates the closed-loop feedback system response
General:	
CONSIS	Performs a consistency check on the number of circuit elements entered by the user

Table 2.3. GCAP analysis options

As an example of a transient analysis, the system response given by equation (2.9) can be separated into the forced (zero-input or steady-state) and natural (zero-state or transient) responses, and studied individually. The circuit as a whole, or any sub-circuit (eg a semiconductor device or any specified group of elements) can be analysed for its response to a transient voltage pulse. The rise-time, fall-time, overshoot, etc, of the resulting waveform can be calculated.

Each analysis contains further sub-divisions. For example, GCAP may be requested to perform a sensitivity analysis in which the values of all components are systematically varied by either a user-specified amount, or 5% (default value), and the circuit studied to see whether it performs within specification limits. Also, the effect of different temperatures on performance can be investigated.

A more complete description of programs as they are applied to various circuits is given in the GCAP user manual[21].

2.4.4 FORMATTING OF I/O DATA

The third and final set of commands consists of structure descriptors. Although stored in an array (STRUCT), this is more correctly a collection of general statements describing the formatting of the I/O data. These include the PRINT and PLOT commands, which are required for tabular listings of data and graphics, respectively. The user can specify which parameters are to be displayed. The voltage across, current through or power dissipated/energy stored in any circuit element can be monitored and output in the form of a printout over a user-specified time interval. Alternatively, a graphics

subprogram provides a visual display of the results. Any additional output data such as comments, headings, etc, are classed as structure descriptors, a complete list of which is presented in Table 2.4.

Descriptor	Function
PRIMAT	Causes GCAP to provide a printout of all matrices
INFO	Provides information on GCAP
TITLE	Assigns a title to the particular analysis which is printed as the heading for each section of output
PRINT	Provides a printout of simulation results over a specified time or frequency range
PLOT	Provides a plot of simulation results over a specified time or frequency range

Table 2.4. GCAP structure commands

2.4.5 PROGRAM FLOW DIAGRAM

A flow diagram outlining the important structural features of the main program is shown in Figure 2.2 and a corresponding representation of the most important subprograms in Figure 2.3. A complete program listing is given in Appendix E.

2.5 CONCLUSIONS

A general circuit analysis program has been designed and developed which has widespread applications to high peak power and repetition rated systems. GCAP represents a pulsed power equivalent of MICROCAP, retaining the full range of analysis options but being applicable also to non-linear and time-varying

circuits. Furthermore, GCAP contains a library of commonly-used devices such as semiconductors or the user may define complicated circuit elements. The size and complexity of the circuit is limited only by the available computer memory. The use of the Pascal programming language increases the scope and versatility of the code which requires no specialised knowledge of the computational method. The code may provide DC, AC, transient and frequency solutions.

Chapter Six contains a detailed example of the application of GCAP to analyse the case of a flux-controlled saturable bypass inductor placed in parallel with the laser head.

REFERENCES FOR CHAPTER 2

1. A. Vladimirescu, K. Zhang, A. R. Newton, D. O. Pederson and A. Sangiovanni-Vincentelli, 'SPICE Version 2G User's Guide', Reference Manual, Cromemco Inc. (1981).

2. MICRO-CAP II Microcomputer Circuit Analysis Program, Reference Manual, Spectrum Software, 1021 South Wolfe Road, Sunnyvale, CA 94086.

3. C. Young, "Modeling pulsed electric discharges using a circuit analysis code", Sixth IEEE Pulsed Power Conference, pp.397-400, Arlington, VA, June 29 - July 1, 1987.

4. V. Bello, "Computer program adds SPICE to switching-regulator analysis", Electronic Design, pp.89-95, March 5, 1981.

5. R. M. Roark, M. E. Parten, L. B. Masten and T. R. Burkes, "Pulse forming networks with time-varying or non-linear resistive loads", IEEE Thirteenth Pulse Power Modulator Symposium, pp.46-51, Buffalo, NY, June 20-22, 1978.

6. D. Putley, "Analysis and modelling of the Culham experimental compulsator", Sixth IEEE Pulsed Power Conference, pp.514-517, Arlington, VA, June 29 - July 1, 1987.

7. B. Z. Hollmann, "Net II simulation of a pulse forming line (PFL) with spark gap and load", Sixth IEEE Pulsed Power Conference, pp.664-667, Arlington, VA, June 29 - July 1, 1987.

8. T. A. Mace, J. W. Gray, R. C. McLachlan and I. Dobson, "CONNIE - A general AC - DC converter and transient circuit analysis program", IEEE Eighteenth Power Modulator Symposium, pp. 109-116, Hilton Head, SC, June 20-22, 1988.

9. R. M. Nelms, S. R. Newton, G. B. Sheble' and L. L. Grigsby, "Simulation of transmission line transients using a personal computer", IEEE Eighteenth Power Modulator Symposium, pp.229-232, Hilton Head, SC, June 20-22, 1988.

10. R. M. Nelms, B. B. Reid and L. L. Grigsby, "Modeling and simulation of power conditioning equipment containing saturable inductors using a personal computer", IEEE Eighteenth Power Modulator Symposium, pp.233-237, Hilton Head, SC, June 20-22, 1988.

11. B. Friedland, 'Control System Design: An Introduction to State-space Methods',Ch. 2, McGraw-Hill International (1987).

12. T. R. Bashkow, "The A matrix, new network description", IRE Trans. on Circuit Theory, vol.CT-4, pp.117-120 (Sept, 1957).

13. P. R. Bryant, "The explicit form of Bashkow's A matrix", IRE Trans. on Circuit Theory, vol.CT-9, pp.303-306 (Sept, 1962).

14. R. D. Middlebrook and S. Çuk, "A general unified approach to modelling switching-converter power stages", IEEE Power Electronics Specialists Conference, pp.18-34, 1976 (IEEE Publication 76CH1084-3 AES).

15. L. O. Chua, C. A. Desoer and E. S. Kuh, 'Linear and Nonlinear Circuits', McGraw-Hill (1987).

16. E. S. Kuh and R. A. Rohrer, "The state-variable approach to network analysis", Proc. IEEE, vol.53, pp.672-686 (July, 1965).

17. R. A. Rohrer, 'Circuit Theory: An Introduction to the State-variable Approach', McGraw-Hill (1970).

18. A. Ralston and P. Rabinowitz, 'A First Course in Numerical Analysis', McGraw-Hill (1978).

19. A. Ralston and H. S. Wilf, 'Mathematical Methods for Digital Computers', Wiley (1966).

20. J. A. Borrie, 'Modern Control Systems: A Manual of Design Methods', Prentice-Hall International, Ch. 3, Section 6, p.170 (1986).

21. A. K. Kidd, 'GCAP Version 01 User Manual', Reference Manual, University of St. Andrews (1989).

22. P. M. Chirlian, 'Analysis and Design of Integrated Electronic Circuits. Volume 1: Semiconductor Devices', Harper & Row Ltd. (1982).

23. L. W. Nagel, "SPICE 2: A computer program to simulate semiconductor circuits", Memorandum No. ERL-M520, College of Engineering, University of California, Berkeley, May, 1975.

FIGURES FOR CHAPTER 2

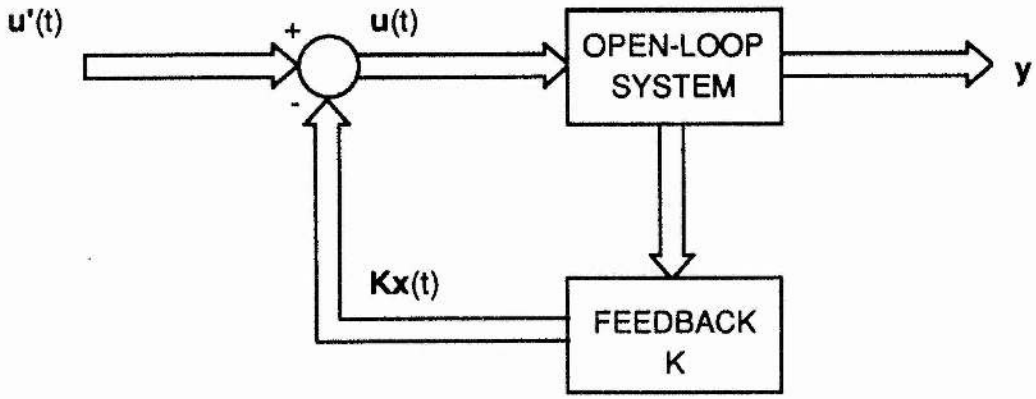


Figure 2.1. Schematic diagram of state feedback

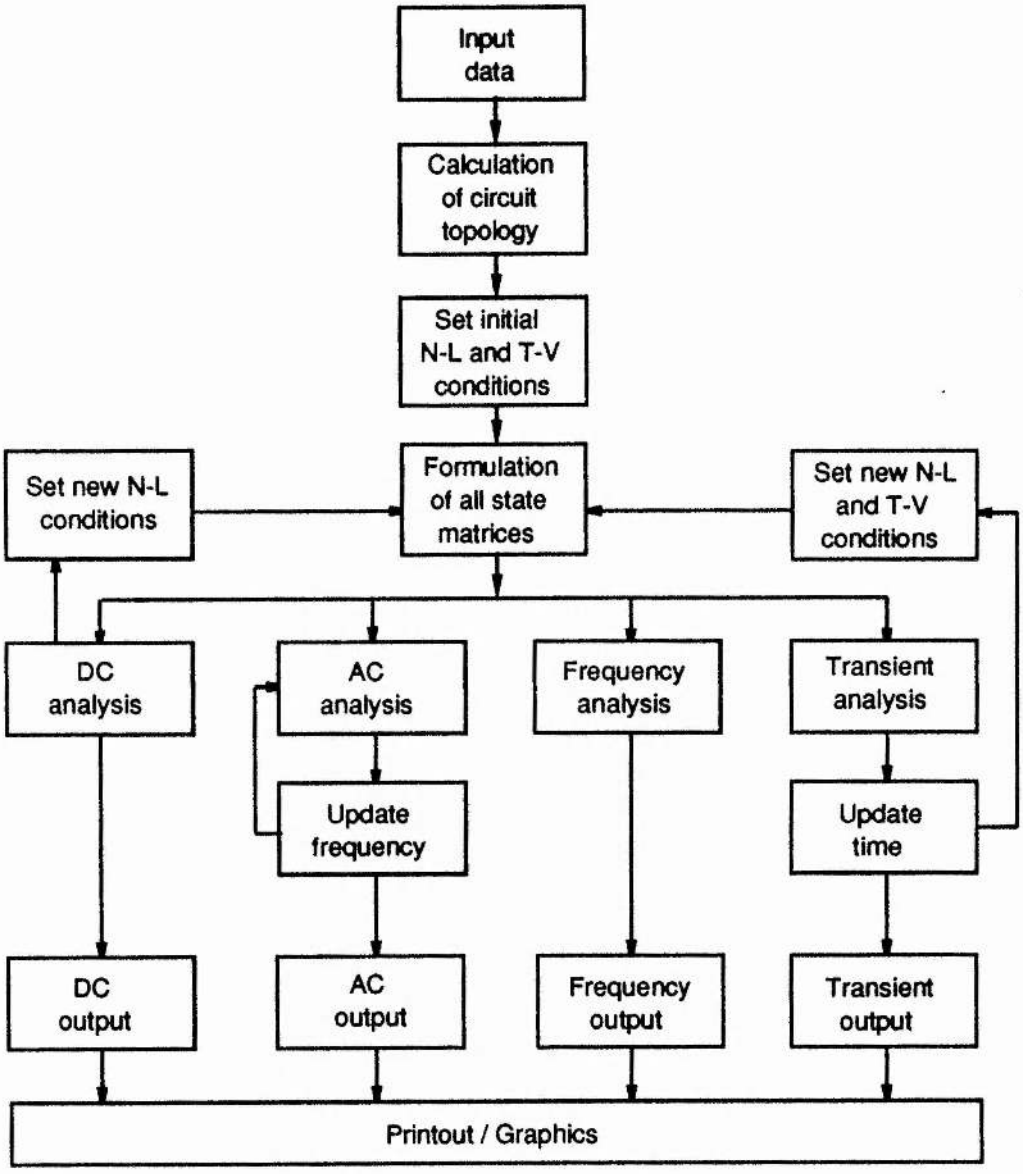


Figure 2.2. GCAP main program flow diagram

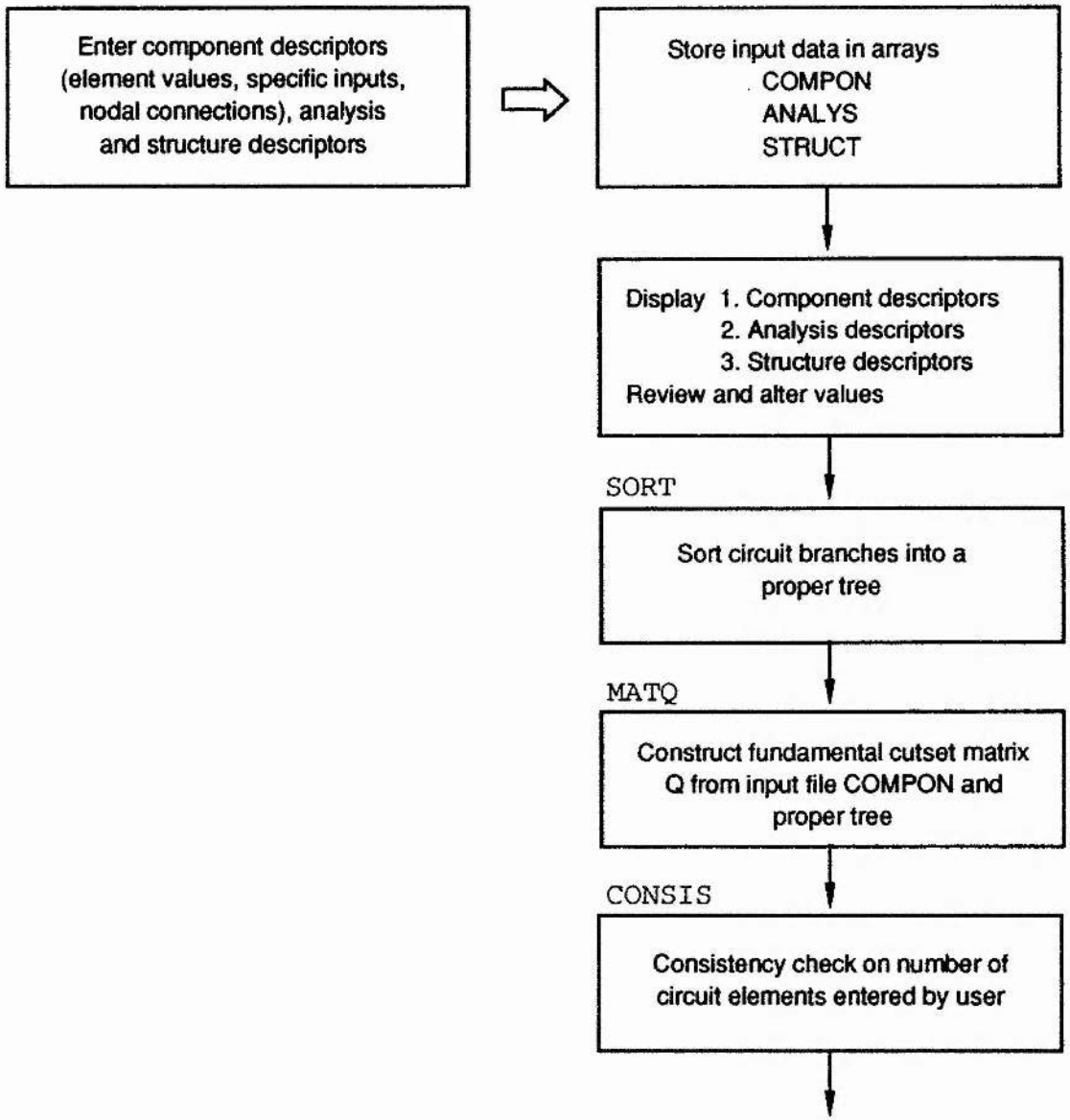


Figure 2.3. GCAP main subprograms flow diagram. Bracketed figures refer to equations in the text

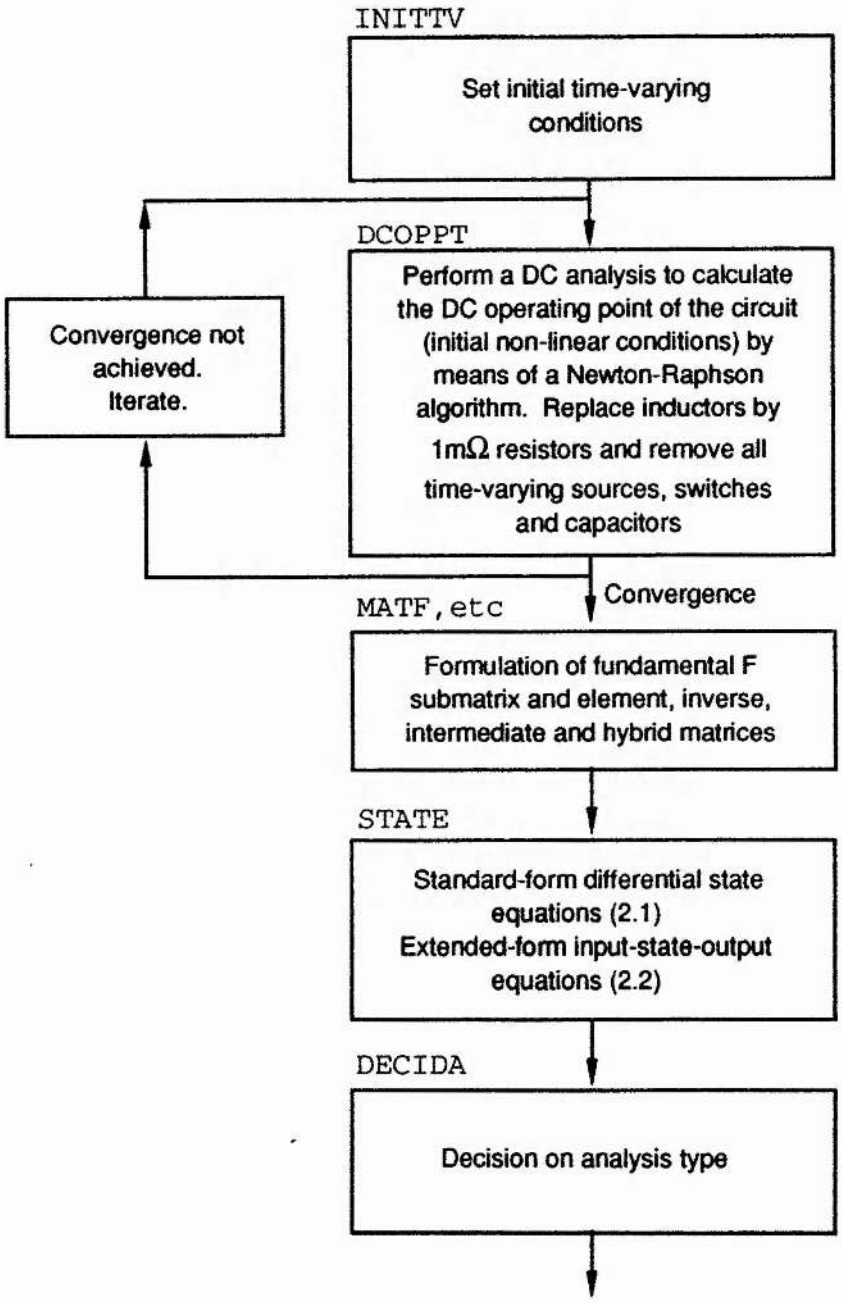


Figure 2.3 (Continued). GCAP main subprograms flow diagram .

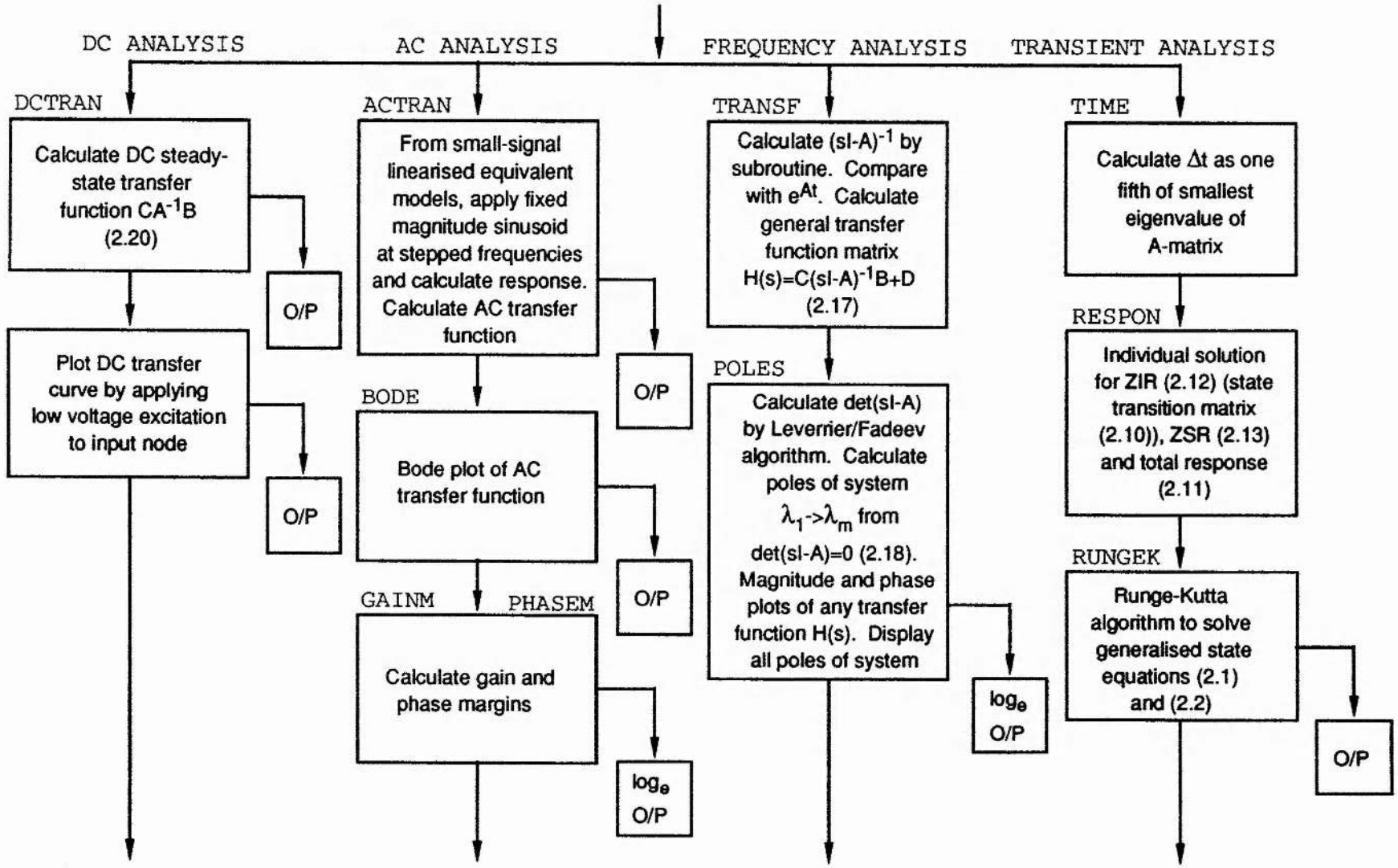


Figure 2.3 (Continued). GCAP main subprograms flow diagram

Figure 2.3 (Continued). GCAP main subprograms flow diagram

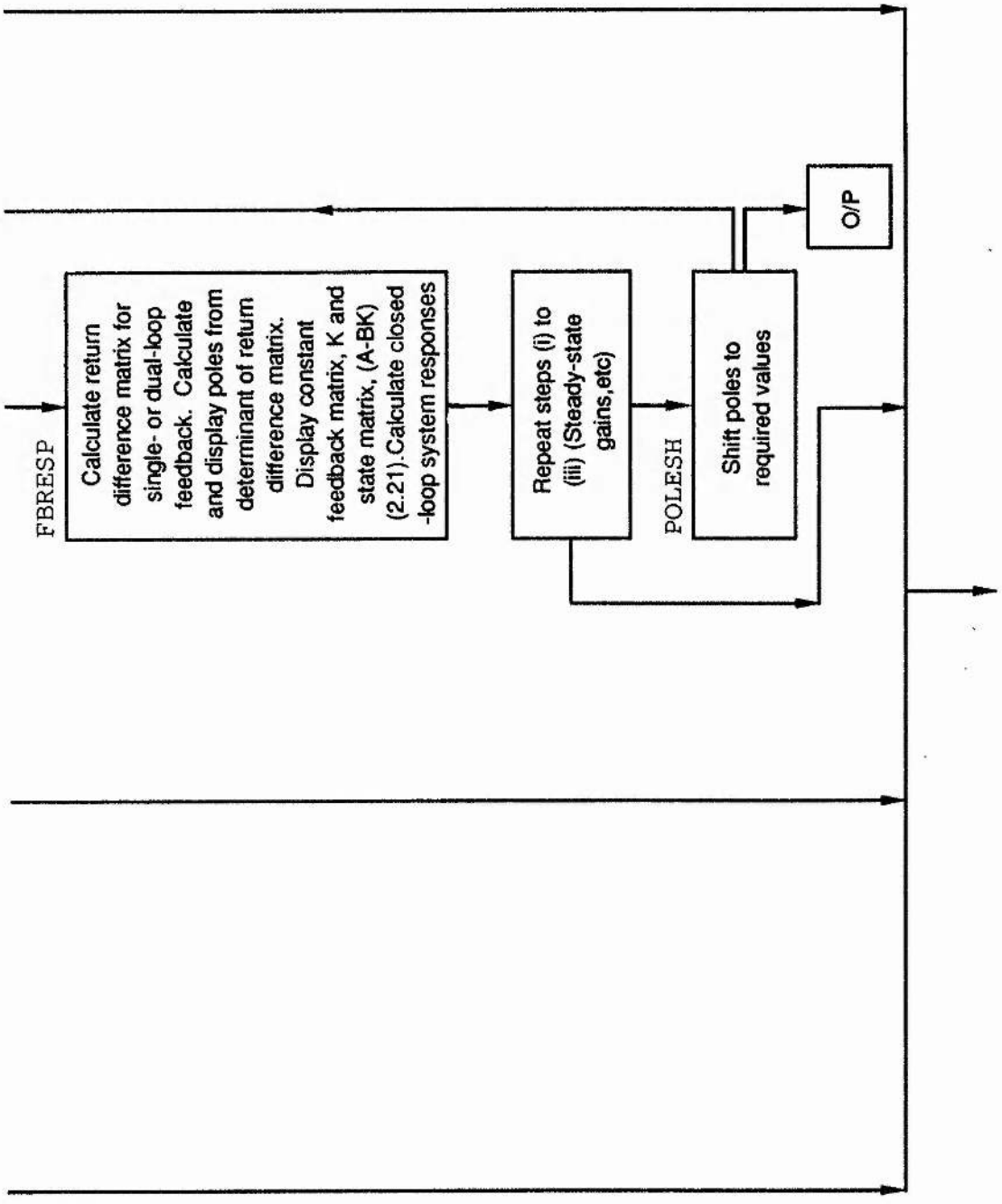


Figure 2.3 (Continued). GCAP main subprograms flow diagram

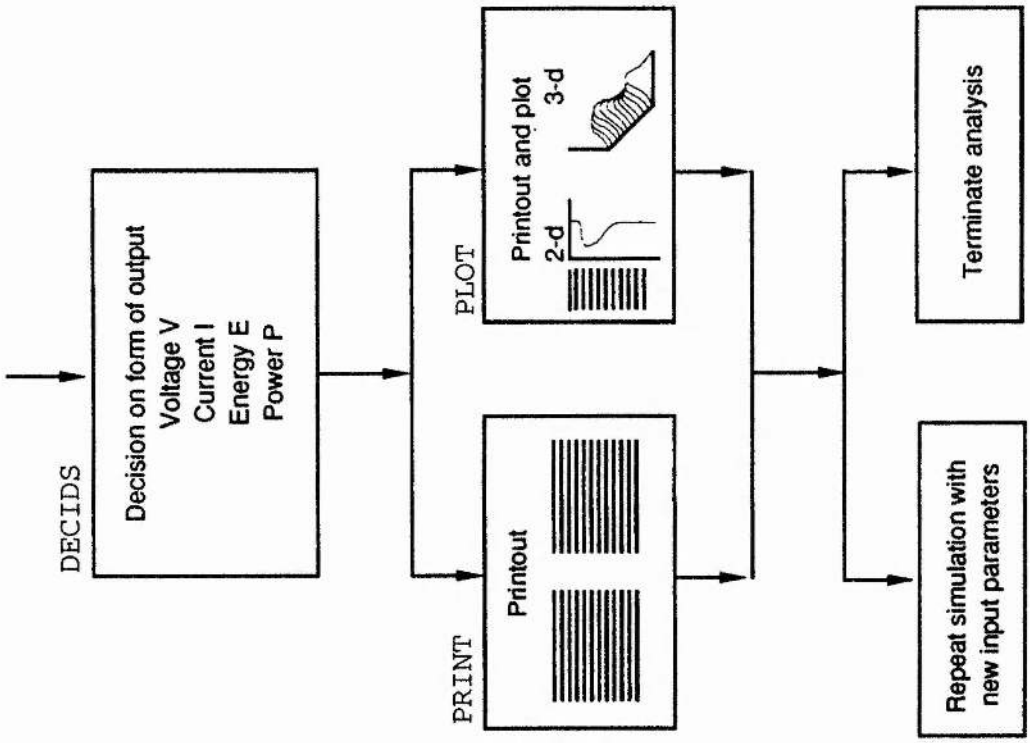


Figure 2.3 (Continued). GCAP main subprograms flow diagram

Chapter 3

Rate Processes in the Strontium Vapour Laser

3.1 INTRODUCTION

The strontium vapour laser (SVL) is a repetitively-pulsed recombination laser; lasing occurs in the afterglow of a discharge current pulse during the recombination of excited state species to the ground state. Experimentally, the interaction between the large number of variable parameters involved makes it difficult to identify the effect of varying a single parameter. For example, the choice of repetition rate affects the number density of excited species available for the next pumping pulse (self-preionisation); although the buffer gas pressure can be specified external to the tube, it varies internally due to transient heating.

A computer program is written to simulate the entire system of the laser and its associated circuitry. The aim in developing this code is to investigate the behaviour of the SVL in response to parametric variation of the electrical discharge circuit. Modelling provides information regarding dominant chemical processes within the plasma and establishes guidelines for the required performance of the modulator. The simulations will be compared with experimental results and used to identify the limitations on performance of recombination lasers (in particular strontium, but the analysis will be applicable also to other recombination laser systems such as barium and calcium, by modification of the relevant atomic and thermodynamic parameters). The

approach described in this thesis is novel for two reasons. Firstly, most of the previously published modelling of the strontium vapour laser has considered a spatially homogeneous plasma and has investigated only temporal developments. However, this approach has difficulty in predicting laser performance since experiments have shown that the volumetric scaling of SVLs is limited by excessive heating of the centre of the discharge[1]. A spatially inhomogeneous treatment is required to explain the observed performance of the strontium laser for discharge tube diameters in excess of approximately 10-15mm. The model described here is zero-dimensional and is applicable to narrow-bore (less than 15mm diameter) strontium lasers. However, the program has the facility to include explicit radial dependence of all kinetic and thermodynamic variables.

The second novel approach is the use of state variables to describe the internal states of the system. These are electrical, atomic, optical and thermodynamic parameters such as voltage, current, number densities, temperature and pressure. Use of state variables provides a detailed insight into the development of the plasma and the characterisation of the system by a set of coupled first order differential equations rather than higher order equations, simplifies their solution. The equations are integrated numerically using the method of finite differences. In this way, a complete picture of the kinetics of the plasma build-up and decay is produced.

Topics related to the modelling of the kinetics of the SVL treated by other authors are discussed in Section 3.2. These include:

(1) zero-dimensional modelling of recombination lasers

(2) pulsed gas discharge lasers

and (3) radial distributions in DC discharges.

The ideas underlying topics (1) to (3) are brought together in Section 3.3 and form the basis of the SVL model. The pulsed SVL system represents a non-equilibrium plasma. The equations describing the time rates of change of the state variables, derived from the Boltzmann collision equation, together with the electrical circuit equations, must be solved simultaneously. The atomic and optical parameters are discussed in Section 3.4 and the thermodynamic parameters in Section 3.5. The electrical equations describing the discharge circuitry allow for non-linear/time-varying elements and are briefly described in Section 3.6. Transport processes (discussed in Section 3.7) play a fundamental role in determining spatial distributions; the transport coefficients are derived from the higher order moments of the Boltzmann collision equation. This chapter concludes with a summary of the state-space description of the laser circuit and discharge parameters.

3.2 PREVIOUS WORK ON RELATED TOPICS OF MODELLING

Recombination lasers were first proposed by D. R. Bates[2] who carried out detailed calculations on optically-thin hydrogen-ion plasmas. This work was also important in that the quasi-stationary period of excited states was identified (see below). Subsequent calculations performed on hydrogen[3,4], lithium[5] and strontium[1,6,7] involving solution of the rate equations for the populations of atomic levels and (for the cases of lithium and strontium) simultaneous solution for electron energy, electron density n_e and temperature T_e, indicate the following general relaxation scheme for free electron densities $n_e \sim 10^{13}$-10^{16}cm^{-3} and temperature $kT_e \sim 0.1$-0.5eV. The electron temperature in the immediate afterglow of a current pulse is of the order of 5-10eV. On rapid termination of the pulse, T_e falls off exponentially until "stationary drainage"

is established, where the number of particles entering a particular energy level equals the number leaving, thus maintaining a constant level population. During this time, T_e is essentially constant. This situation arises because the characteristic relaxation times of free electrons greatly exceeds the average time of collisional de-excitation between levels. During this time, the net flow of particles into a given atomic/ionic state is zero,

$$\frac{\partial n_e}{\partial t} = - \frac{\partial N_{i>1}}{\partial t} = 0 \qquad (3.1)$$

ie populations of all levels are constant except for the ground states of atoms and ions which are slowly filled/emptied. The relaxation times for establishing a stationary population have been calculated for hydrogen-like ions[6]. Results obtained for hydrogen by merely substituting various values of n_e and T_e into the "stationary flow" rate equations, show that for $n_e \sim 10^{13}$-10^{15} cm^{-3}, on rapid cooling from $kT_e \sim 2eV$ to 0.1eV, a population inversion is achieved for a number of levels. Cooling can be effected in two stages: after a time ~ 10^{-9}-10^{-8} seconds from 2-3 to 0.2-0.5 eV, and then after a time ~ 10^{-8}-10^{-7} seconds down to 0.05-0.1 eV. Calculations for lithium[5] attribute the appearance of a population inversion in a dense recombining plasma to the unilateral recombination electron flux. This immediately suggests the possibility of obtaining large gain (large population inversions) by increasing the electron density.

The models described above have assumed values of electron temperature and density in the immediate afterglow of the discharge current pulse, have taken no account of discharge circuitry and have considered spatially homogeneous plasmas. Computer predictions based on this "zero-dimensional" model have proved inadequate to explain the radial and temporal evolution of the events

occurring in these plasmas. Recently, a more detailed model of recombination lasers has been published[8]. Extensive work has been carried out on the copper vapour[9,10] and copper chloride[11] lasers. Although the lasing mechanism of the CVL is different to that of the SVL, Kushner[9,10] provides some useful ideas on discharge and afterglow processes. Kushner[9] proposes a self-consistent model in which, by comparing the initial and final values of population densities and electron temperature over one complete period of the discharge cycle, the initial values are modified and the simulation is repeated until the initial and final values converge to a common value.

Several authors have studied the spatial distribution of various parameters in DC discharges[12-17]. The basic model, consisting of a system of one kind of neutrals only, was proposed by Schottky[12,13]. Electron, ion and neutral gas temperatures are considered to be spatially uniform. However, this assumption breaks down with increasing discharge current. The temporal and radial variations of neutral gas density and temperatures in DC discharges even at low currents (<10A) have been found to be strongly temperature-dependent[14] and there is experimental evidence for radial temperature variations at high currents (>50A)[15]. The studies of Eden[17] on radial profiles showed strong depletion of neutral species on axis.

3.3 DESCRIPTION OF THE MODEL

The aims of the model introduced in this thesis are as follows:

(1) To simulate both the discharge pulse and afterglow processes in the plasma.

(2) The model must be able to treat processes described by second and higher-order differential equations in a state-variable form.

(3) The ratio of helium to neon in the buffer gas should be variable.

(4) The code should be applicable to other recombination lasers by alteration of the relevant atomic, optical and thermodynamic parameters.

The model is based on an infinitely long plasma column of radius R confined by insulating walls (Figure 3.1). The plasma has five (seven) components consisting of electrons and helium (neon) and strontium neutrals and ions. Electrons and ions are accelerated in the axial direction z under the influence of a longitudinal electric field E_z. The column is axially homogeneous. A uniform longitudinal electric field is assumed. Mathematically, the plasma is completely described by electrical, atomic, optical and thermodynamic parameters. These are then cast as the state variables x_n. This approach has two advantages over conventional methods of plasma modelling:

(1) The equations can always be written as a coupled set of first-order differential equations. By solving these equations numerically, the state matrices may be updated at each successive time step. This allows non-linear and time-varying characteristics to be included. Furthermore, state variable analysis provides a powerful method of manipulating the results obtained by analysis of the A matrix.

(2) The plasma processes can be directly linked to the generalised circuit analysis program (GCAP) described in Chapter Two.

If all particle temperatures are considered to be constant, then the coefficients of the particle balance equation are constant. However, on dropping the unrealistic assumption of constant particle temperatures across the discharge tube cross-section to include explicit radial dependence of each of the state variables, the situation becomes considerably more complicated. The coefficients become radially and temporally dependent, and the electron temperature becomes a function of the neutral gas concentration and the electric field. Furthermore, constant pressure and a low degree of ionisation cannot be assumed in a pulsed gas discharge at high peak currents. As the degree of ionisation of a pulsed plasma varies with time, so too do its properties such as thermal conductivity which is a function of electron temperature, its spatial and temporal gradients, the degree of ionization and the discharge geometry.

Calculations described in this chapter extend the two-fluid model of Ecker and Zoller[16] for a DC discharge to apply to pulsed systems. This necessarily introduces non-linearities of second and higher orders. All state variables describing an inhomogeneous plasma are allowed to be spatially and temporally dependent. The state variables are governed by a system of coupled differential equations with variable coefficients whose solution is not obtainable by conventional methods. The state equations describing the atomic, optical, thermodynamic and electrical processes are derived in turn below.

3.4 RATE EQUATIONS FOR THE SVL

The gas dynamics are completely described by the Boltzmann collision equation[18]

$$\frac{\partial f}{\partial t} + \underline{v}.\underline{\nabla}_r f + \frac{\underline{F}}{m}.\underline{\nabla}_v f = \left[\frac{\partial f}{\partial t}\right]^{\text{collision}}_{\text{radiation}} \tag{3.2}$$

where f is the velocity distribution function and $\underline{\nabla}_r$ is the spatial Laplace operator.

The zero-order moment solution of (3.2) yields the continuity equation for the total number density of species n_n of the plasma[18],

$$\frac{\partial n(\underline{r},t)}{\partial t} + \underline{\nabla}_r.(n_n\langle\underline{v}_n\rangle_r) + \frac{1}{r}\, n_n \langle\underline{v}_n\rangle_r = \left[\frac{\partial n_n}{\partial t}\right]^{\text{collision}}_{\text{radiation}} \tag{3.3}$$

where $\langle\underline{v}_n\rangle_r$ is the radial velocity component (the number average velocity).

The right hand side of the equation (3.3) is considered first. Rate equations for the populations of discrete levels can be concisely written[19]

$$\left[\frac{\partial n_n}{\partial t}\right]^{\text{collision}}_{\text{radiation}} = \sum_{n<n_1}\left(K_{nm}N_m + D_n\right) \tag{3.4}$$

where N_n is the population of the nth level, K_{nm} is the relaxation matrix, D_n represents the arrival of particles from the continuum and n_1 is the effective limit of the continuous spectrum.

The relaxation matrix is given by

$$K_{nm} = U_{nm}n_e + A_{nm} \tag{3.5}$$

where $U_{nm} = \langle \sigma_{nm} v_e \rangle$ is the rate of the transition $n \rightarrow m$ as a result of inelastic collisions with electrons and A_{nm} is the rate of spontaneous transitions $n \rightarrow m$.

Finally, we have

$$\frac{dN_n}{dt} = - n_e N_n \left(\sum_{m>n} V_{nm} + \sum_{m<n} R_{nm} \right) + n_e N_m \left(\sum_{m>n} R_{mn} + \sum_{m<n} V_{mn} \right) + n_e^3 B_{\varepsilon' n}$$

$$- n_e N_n B_{\varepsilon n} - N_n \sum_{m<n} A_{nm} + N_m \sum_{m>n} A_{mn} + n_e^2 A_{\varepsilon n} \qquad (3.6)$$

where V_{nm} and R_{nm} are proportional to the probabilities of non-radiative transitions within the atom, ie inelastic electron-ion collisions. If $E_n > E_m$, then it is a superelastic collision, de-exciting the atom and increasing the electron energy. If $E_n < E_m$, the atom is further excited. The terms $B_{\varepsilon n}$ and $B_{\varepsilon' n}$ are proportional to the probabilities of non-radiative (three-body) recombination and its inverse, electron-collisional excitation, respectively. The quantities A_{nm} and A_{mn} are proportional to the probabilities of radiative spontaneous transitions between levels n and m. The quantity $A_{\varepsilon n} = \sum_{m<n} A_{nm}$ is proportional to the total probability of radiative decay of the state n.

The individual reaction processes between particles in a non-equilibrium plasma are described by the rate equations (3.6). Photon densities are treated in a similar manner. The collisional and radiative processes are shown in Figure 3.2 and a brief description of these follows.

3.4.1 RATE EQUATION PROCESSES

The processes considered in the model are outlined below. In these

equations, A, A^*, A_m and A^+ are, respectively, any neutral, excited, metastable and ionised species (helium, neon or strontium).

3.4.1.1 Spontaneous Radiative Transitions to all Levels Lying Lower in Energy

Experimental data on the probabilities of radiative transitions in the spectrum of SrII is insufficient for reliable estimates. However, estimates can be obtained by the method of Bates and Damgaard[20], which for simple systems such as alkali-ion types (which includes the singly-ionised alkaline-earth metals) gives accurate results. A very brief account of the method follows.

The transition probability per second A_{mn} associated with a spectral line is given by

$$A_{mn} = 2.02 \times 10^{18} \frac{1}{g_n \lambda_{mn}^2} S \qquad (3.7)$$

where g_n is the statistical weight of the upper level involved, λ_{mn} is the wavelength of radiation emitted/absorbed in the transition between levels m and n and S is the line strength (the latter two quantities being expressed in atomic units ($a_o^2 e^2$)).

The line strength can be written in the form

$$S_{nm} = I(M)\ I(L)\ \sigma^2 \qquad (3.8)$$

where I(M) is the relative multiplet strength, depending on the particular multiplet of the transition array, I(L) is a factor depending on the particular line within that multiplet and σ^2 is the square of the one-electron matrix

component. It is a function of the total (n) and orbital (l) quantum numbers of the transition electron, and is a constant for a given transition array:

$$\sigma = \frac{-e}{(4l^2-1)^{0.5}} \int_0^\infty R(n,l)\, R(n',l')\, r\, dr \qquad (3.9)$$

where l is the greater of the two azimuthal quantum numbers involved in the transition $(n,l) \rightarrow (n',l')$ and $\frac{R(n,l)}{r}$ and $\frac{R(n',l')}{r}$ are the normalised initial and final radial eigenfunctions of the active electron.

The total line strength of a multiplet is the sum of the electric dipole moments for that transition:

$$S' = \sum_{mn} D_{mn} = I(M)\, \sigma^2 \quad ; \quad \sum I(L) = 1 \quad . \qquad (3.10)$$

The term I(M) can be found from the two sets of tables published by Goldberg[21,22]. The former uses the Kronig formulae[23] which apply to systems in which the electron transition is of the sort $\gamma+\alpha=\gamma+\beta$, where α and β are one-electron configurations between which radiative transitions are allowed, and neither α nor β contain electrons equivalent to any in γ. Such a transition is the 6s-5p laser transition in SrII, where α=6s, β=5p and γ=0. The second table supplements the first; relative values of strengths of multiplets are converted to absolute strengths in terms of σ^2 by multiplying by a scaling factor *a*.

The term I(L) can be obtained either from the tables of Russell[24] or from those of White and Eliason[25]. In these tables, the relative intensities of various transitions within a multiplet are given on a scale of 1 to 100 for the strongest line in the multiplet, grouped according to multiplicity (2S+1). The relevant factor I(L) is found by reducing this scale to unity. The sum rule[26]

is then used to proportion the relative strengths.

Thus, to calculate the line strength S, it remains only to evaluate the parameter σ^2. The Coulomb approximation may be used for all transitions in the lighter simple systems, but accuracy suffers in certain transitions in the heavier simple systems. The lighter systems include BeII, MgII and CaII, but not SrII or BaII. Instead, the following method will be used.

Bates and Damgaard[20] showed that in the transition $(n^*,l-1) \rightarrow (n^*,l)$, where n^* is the effective principal quantum number, the integral of equation (3.9) can be expressed in the form

$$\frac{3n^*}{2C} = \left[n^{*2}-l^2\right]^{0.5} \qquad (3.11)$$

where C is the excess charge on the nucleus when the active electron is removed.

Thus, we have

$$\sigma(n^*_{l-1},l-1;\ n^*_l,l;\ C) = \frac{1}{C} F(n^*_l,l)\ I(n^*_{l-1},n^*_l,l) \qquad (3.12)$$

with

$$F(n^*_l,l) = \frac{3n^*_l}{2}\left[\frac{n^{*2}-l^2}{4l^2-1}\right]^{0.5} \qquad (3.13)$$

and

$$I(n^*_{l-1},\ n^*_l,\ l) = \frac{2C}{3n^*_l}\ \frac{1}{(n^{*2}-l^2)^{0.5}} \int_0^\infty R(n^*_{l-1},l-1,C)\ R(n^*_l,l,C)\ r\ dr\ . \qquad (3.14)$$

Bates and Damgaard have tabulated values of $F(n^*_l,l)$ and $I(n^*_{l-1},n^*_l,l)$ for

specific n_l^*, n_{l-1}^* and l[20].

Finally, n^* is defined as

$$n^* = \frac{C}{E^{0.5}} \tag{3.15}$$

where, for systems such as SrII with a single electron outside closed shells, E is the energy required to remove the electron from the particular level concerned. The magnitude of E is found in the detailed tables of term values[27].

The Bates and Damgaard method is applied to estimate the spontaneous decay probabilities of the transitions of interest in SrII listed in Table 3.1. Where available, experimental results[28] are provided for comparison. The agreement between theory and experiment is good for the 5^2P-5^2S transitions. The agreement is, however, less satisfactory for the 5^2P-4^2D transitions, and the experimental values are used (where available) in the SVL model described in Chapter Four. Finally, experimental values for the spontaneous decay probabilities of the 6^2S levels appear not to have been published and theoretical values are included in the model for these transitions.

The lower laser level of the 430.5nm transition, $5^2P_{3/2}$, can spontaneously decay to either the $4^2D_{3/2,5/2}$ metastable levels, or to the SrII ground state $5^2S_{1/2}$. From the table, although the transition $5^2P_{3/2}$-$4^2D_{5/2}$ is optically allowed, the probability is low due to the small separation in energy of the levels ($\Delta\upsilon = 7679.8cm^{-1} \Rightarrow \Delta E \approx 0.95eV$). The spontaneous transition to the $4^2D_{3/2}$ level is even less likely. Although the transition to the ground state is more probable ($A_{nm} = 146 \times 10^6 s^{-1}$), another mechanism is necessary to clear

the lower laser level.

Transition	λ_{nm}(Å)	$A_{nm}(x10^6 sec^{-1})$ Theory	Expt.	f_{ik} Theory	Expt.	σ^2 Theory	Expt.
$5^2P_{3/2}$-$5^2S_{1/2}$	4078	146	143±6	0.737	0.71±0.03	4.93	4.8±2
$5^2P_{1/2}$-$5^2S_{1/2}$	4216	120	127±5	0.31	0.34±0.015	4.27	4.8±0.2
$5^2P_{3/2}$-$4^2D_{5/2}$	10327	6.9	8.7±1.5	0.074	0.096±0.02	0.417	0.51±0.09
$5^2P_{3/2}$-$4^2D_{3/2}$	10040	3.5	1.0±0.2	0.053	0.016±0.03	0.347	0.51±0.09
$5^2P_{1/2}$-$4^2D_{3/2}$	10905	9.5±2	(a)	(b)	0.084±0.02	(b)	0.61±0.12
$6^2S_{1/2}$-$5^2P_{3/2}$	4305	130	140	0.180	(a)	2.56	(a)
$6^2S_{1/2}$-$5^2P_{1/2}$	4162	71	65	0.093	(a)	2.28	(a)

Table 3.1. Spontaneous transition probabilities A_{nm}, oscillator strengths f_{ik} and σ^2 in SrII

Notes: (a) Not available in published literature
(b) Experimental estimate considered to be more accurate

3.4.1.2 Electron-impact Processes

(i) Non-radiative Transitions Within the Atom

During the recombination non-equilibrium period, which is characterised by high values of n_e and low values of T_e, the high plasma density means level populations are affected significantly by inelastic collisions of slow electrons:

$$A + e^- \rightarrow A^* + e^- \quad (3.16)$$

$$A^* + e^- \rightarrow A + e^- \quad (3.17)$$

where A^* is either a metastable or excited state.

Denoting the rate of such electron excitation/de-excitation from level 1 to level 2 as F_{12}, then, we have

$$F_{12} = \langle \sigma_{12} v_e \rangle \qquad cm^3 sec^{-1} \tag{3.18}$$

where σ_{12} is the cross-section of the transition $1 \rightarrow 2$ by electron-impact excitation/de-excitation and $\langle \sigma_{12} v_e \rangle$ is the product of the cross-section and the electron velocity averaged over this velocity. The electron velocity is assumed to be Maxwellian.

The collisional rate coefficients $\langle \sigma(v)v \rangle_k^l$ for collisions between particles of kind k and l are given by

$$\langle \sigma(v)v \rangle_k^l = \int_v \sigma_k^l(w_k^l)\, f_k^l(w_k^l)\, w_k^l \, dw$$

$$\int_w f_k \, dw = 1 \tag{3.19}$$

where $f_k(w_k^l)$ represents the velocity distribution function between particles of kind k and l as a function of their relative velocity w_k^l, and σ_k^l is the cross-section as a function of w_k^l.

There appears to be no published data for the cross-sections of the collisional processes in SrII. We shall use the modified Born cross-sections, calculated together with the products $\langle \sigma v_e \rangle$ from the data of Sobel'man[29].

Effective cross-sections for elastic collisions resulting in an excitation process between levels 1 and 2 can be obtained from the Born formula

$$\sigma_{12} = \pi a_o^2 \left[\frac{R_y}{\Delta E}^2\right] \left[\frac{E_2}{E_1}^2\right] \frac{Q_{\chi min}}{2 l_1 + 1} \Phi(u) \qquad (3.20)$$

where πa_o^2 is the atomic unit of cross-section, $\Delta E/R_y$ is the energy of the transition in Rydbergs, E_1 and E_2 are the energies of levels 1 and 2, Q_χ is a factor dependent on the angular quantum numbers l_1 and l_2 and u is the energy of the scattered electron.

The term $\Phi(u)$ is a function of parameters C and ϕ which are determined by the method of least squares using a method analagous to that of Bates and Damgaard for oscillator strengths, and listed in tables[29].

At high energies (E_1 » (E_2-E_1)), the Bethe formula[29] implies that for optically allowed transitions, the order of magnitude of the maximum value of σ_{12} is given by[30]

$$\sigma_{max} \simeq \frac{2f_{12}}{(\Delta E/R_y)^2} .\pi a_o^2 \qquad (3.21)$$

Assuming a Maxwellian distribution of electrons, that is

$$f(v) = n_e \left[\frac{m}{2\pi k T_e}\right]^{3/2} \exp\left[-mv^2/2kT_e\right] \qquad (3.22)$$

and substituting equations (3.21) and (3.22) into (3.19) gives[31]

$$\langle\sigma_{12}v_e\rangle = \left[\frac{8e}{m\pi}\right]^{0.5} \sigma_{max} \left[\frac{kT_e}{e}\right]^{0.5} \left[1+\frac{\varepsilon_o}{kT_e}\right] \exp\left[\frac{-\varepsilon_o}{kT_e}\right] \quad .(3.23)$$

The rate of collisional de-excitation is found from the principle of detailed balance, according to which, the rate of production of N_2 from N_1 (V_{12}) exactly equals the rate of production of N_1 from N_2 (R_{21}), so we get

$$V_{12} = n_e N_1 \langle\sigma_{12}v_e\rangle = n_e N_2 \langle\sigma_{21}v_e\rangle = R_{21} \quad . \qquad (3.24)$$

Thus, combining the Boltzmann distribution

$$N_2 = N_1 \frac{g_1}{g_2} \exp\left[\frac{-\Delta E}{kT_e}\right] \quad , \qquad (3.25)$$

where g_1 and g_2 are the degeneracies of the lower and upper levels, respectively, with equation (3.24), provides the following expression for the reverse (relaxation) transition 2→1 due to elastic collisions

$$\langle\sigma_{21}v_e\rangle = \frac{g_1}{g_2} \exp\left[\frac{\Delta E}{kT_e}\right] \langle\sigma_{12}v_e\rangle \quad . \qquad (3.26)$$

Then, substituting equation (3.21) for σ_{max} into equation (3.23) reveals that the rate of electron excitation from level 1 to 2 is given by

$$\langle\sigma_{12}v_e\rangle \alpha f_{12} \frac{1}{(\Delta E_{21})^2} (kT_e)^2 \left[1+\frac{\Delta E_{21}}{kT_e}\right] \quad . \qquad (3.27)$$

During the recombination non-equilibrium period, which is characterised

particularly by the high values of n_e and low values of T_e, we have $\Delta E_{21} \gg kT_e$ and equation (3.27) becomes

$$\langle \sigma_{12} v_e \rangle \;\alpha\; f_{12} \frac{1}{\Delta E_{21}^{1/2} T_e} \exp\left[\frac{-\Delta E_{21}}{kT_e}\right] \quad . \tag{3.28}$$

Similarly, we get

$$\langle \sigma_{21} v_e \rangle \;\alpha\; f_{12} \frac{g_1}{g_2} \frac{1}{\Delta E_{21}^{1/2} T_e} \quad . \tag{3.29}$$

Finally, from equation (3.26), for a fixed pair of energy levels, we obtain

$$\frac{\langle \sigma_{12} v_e \rangle}{\langle \sigma_{21} v_e \rangle} \;\alpha\; \exp\left[\frac{\Delta E_{21}}{kT_e}\right] \quad . \tag{3.30}$$

Thus, when n_e is high enough and the separations between levels in a particular group are less then, or comparable with, kT_e, the probabilities of direct collisional transitions within a group exceed the probabilities of optical transitions both within the group and to lower groups. Therefore, the populations of the levels within a group obey a Boltzmann distribution with an electron temperature T_e. Thus, the 6S-5P transition in SrII is ideal for establishing a population inversion since the 6S level is the lowest in one group and 5P is the highest in another (see Figure 3.2)

(ii) Ionisation and Recombination Processes

Neglecting, for the moment, transport in space, the rate of change of the electron concentration is

$$\left(\frac{\partial n_e}{\partial t}\right)^{\text{collision}}_{\text{radiation}} = \sum_{nq} (N_n W^q_{ne} - n_e W^q_{en}) \qquad (3.31)$$

where W^q_{ne} and W^q_{en} represent the ionisation and recombination probabilities.

For times which are not too short after the end of the excitation pulse, the quasi-stationary approximation[19] allows the system of equations (3.31) to be reduced to only one equation:

$$\left(\frac{\partial n_e}{\partial t}\right)^{\text{collision}}_{\text{radiation}} = N_1 n_e \beta - n_e^3 \alpha \qquad (3.32)$$

where α and β are the generalised recombination and ionisation coefficients.

(a) Recombination

The generalised recombination coefficient accounts for collisional and radiative recombination:

$$\alpha = \alpha_c + \alpha_r \qquad . \qquad (3.33)$$

The collisional process[32]

$$A^+ + e^- \rightarrow A + h\upsilon \qquad (3.34)$$

is negligible at high electron densities[33]. Instead, the following three-body collisional-radiative recombination process dominates[34]:

$$A^+ + e^- + e^- \rightarrow A + e^- \quad . \tag{3.35}$$

Calculation of α (and β) can be by numerical methods (eg see Bates et al[2]). The problem is complicated and is considered in the review paper by Biberman et al[35]. Using the "modified diffusion approximation", the following was derived:

$$\alpha_r = \frac{h^3 e^4}{\pi m^2 R_y^3} \cdot \frac{2\Lambda}{3\pi^{1/2}} \cdot \left[\frac{R_y}{T_e}\right]^{9/2} \tag{3.36}$$

where Λ (= 0.2) is the mean value of the Coulomb logarithm for the excited states. This $T_e^{-9/2}$ dependence holds true irrespective of the type of atom considered, but for $kT_e \geq 0.25$ eV, deviations from this relationship begin. For hydrogen-like atoms and ions, it is true for a considerably greater range of T_e and the diffusion approximation is applicable. Several authors (see for example[34,36]) confirm this dependence of α_r on n_e and T_e for singly-charged ions in a cold, dense plasma. For multi-charged ions, Veselovskii[33] derived the following:

$$\alpha_r = 1.8 \times 10^{-8} z^3 L T_e^{-9/2} \tag{3.37}$$

where $L = \log_e (z^2+1)^{1/2}$.

In conclusion, in a dense plasma at low temperatures, the triple recombination coefficient is governed by inelastic collisions and equation (3.37) tells us that the rate of this recombination increases strongly with decreasing T_e and increasing n_e and charge number on the recombining ions, z.

Calculations for a fully-ionised hydrogen gas[37] show that the maximum contribution to the recombination rate comes from states with principal quantum numbers n between 5 and 8. For $n \geq 10$, the contribution is negligible. Hinnov and Hirschberg[38] extended the study to helium and concluded that the high-energy states ($n>5$) are populated by three-body electron-ion recombination and electron-impact ionisation. Radiative cascading from higher levels contributes negligibly to the recombination coefficient (except at low T_e and n_e).

(b) Ionisation

The general formula for electron-impact ionisation of an atomic species A is

$$A + e^- \rightarrow A^+ + e^- + e^- \quad . \tag{3.38}$$

The parameters a and b are related by the ionisation-equilibrium constant[29]

$$\langle \sigma^i v_o \rangle = 2 \frac{g_1}{g_o} \left[\frac{2\pi m k T_e}{h^2} \right]^{3/2} \exp \left[\frac{-I}{kT_e} \right] \langle \sigma^r v v_r \rangle \tag{3.39}$$

where we have $b = \langle \sigma^i v_o \rangle$, $a = \langle \sigma^r v v_r \rangle$, g_o and g_1 are the statistical weights of the ground and excited states, respectively, and I is the ionisation energy.

Hence, combining equations (3.37) and (3.39), we get

$$\beta = 5.5 \times 10^{14} \frac{g_1}{g_o} T_e^{-3} \exp \left[\frac{-I}{kT_e} \right] \quad . \tag{3.40}$$

3.4.1.3 Atomic and Molecular Collisions

(i) Dissociative Recombination of He_2^+ and Ne_2^+

The molecular ion A_2^+ is formed by the reaction

$$A^+ + A + B \rightarrow A_2^+ + B \qquad (3.41)$$

in which the species A and B may be either helium or neon atoms.

Although at a few hundred millibar, milliseconds are required for significant production of A_2^+, the process is important in considering the afterglow. At these pressures, the dissociative recombination process

$$A_2^+ + e^- \rightarrow A_2^{'} \rightarrow A^* + A^* \qquad (3.42)$$

dominates over three-body recombination[39,40]. The species $A_2^{'}$ is an intermediate unstable molecule.

(ii) Collisional De-activation

Collisional de-activation is represented by

$$A_m + B \rightarrow A + B \qquad (3.43)$$

(iii) Energy Exchange Involving Metastable or Ionised Helium/Neon

Penning collisions: $$A_m + B \rightarrow A + B^+ + e^- \quad (3.44)$$

$$A_m + B_m \rightarrow A^+ + B + e^- \quad (3.45)$$

Charge exchange: $$A^+ + B \rightarrow A + B^+ \quad (3.46)$$

$$A^+ + B \rightarrow A + B^{++} + e^- \quad . \quad (3.47)$$

The second and third terms in equation (3.3) require evaluation of the convective velocity of particles, $\langle \underline{v}_m \rangle_r$. The transfer rate of momentum is obtained from the first moment solution to the Boltzmann equation:

$$\left[\frac{\delta n_{k.i} m_k \langle \underline{w}_{k.i} \rangle}{\delta t} \right]^{\text{collision}}_{\text{radiation}} = \left[\frac{\delta n_{k.i} m_k \langle \underline{w}_{k.i} \rangle}{\delta t} \right] + \underline{\nabla}_r \cdot \left[n_{k.i} m_k \langle \underline{w}_{k.i} \underline{w}_{k.i} \rangle \right] - n_{k.i} \underline{F}_{k.i} \quad (3.48)$$

Electrons and ions are required to flow with equal radial velocities ($\langle \underline{v}_i \rangle_r = \langle \underline{v}_n \rangle_r$) to maintain charge neutrality. If electron transport in the radial direction is assumed to be limited by ambipolar diffusion, then the convective velocity is given by

$$\langle \underline{v}_i \rangle_r = \langle \underline{v}_n \rangle_r - \frac{D_a}{n_e T_e} \underline{\nabla}_r (n_e T_e) \quad . \quad (3.49)$$

3.4.1.4 Optical Transitions

Since the SVL model described here is approached from a kinetics (as opposed to optics) standpoint, it is sufficient to describe the development of the laser

intensity in terms of the photon density N_υ per unit frequency. This approach is justified at the high pressures encountered in the SVL where broadening collisions are sufficiently frequent to destroy phase information.

The rate of change of photon density is

$$\frac{dN_\upsilon}{dt} = \left[\frac{L}{L_c} \frac{A_{21} c^3}{8\pi\upsilon_{21}^2} \left[N_2 - \frac{g_2}{g_1} N_1 \right] g(\upsilon_o) - c\left[\frac{(1-R)}{L_c} - \theta \right] \right] N_\upsilon$$

$$+ N_2 A_{21} \cdot \frac{d\Omega}{4\pi} \qquad (3.50)$$

where L and L_c are the length of the active medium (electrode separation) and optical cavity (mirror separation); L/L_c is the fraction of the cavity transit time for which a photon interacts with the excited atomic energy levels. Subscripts 1 and 2 refer to the lower and upper laser levels and A_{21} is the spontaneous transition probability between these two levels. The reflectivity of the output mirror is $\mathscr{R}$, and θ represents the parasitic cavity losses per unit length. The factor $d\Omega/4\pi$ represents that fraction of spontaneously emitted photons that are emitted within the correct solid angle to be amplified. The term $g(\upsilon_o)$ is the line shape function which for a homogeneously broadened transition is given by

$$g(\upsilon_o) = \frac{2}{\pi\Delta\upsilon_o} \qquad (3.51)$$

where $\Delta\upsilon_o$ is the transition linewidth.

Radiation trapping is insignificant due to the predominance of collisional transitions in determining strontium energy level populations. For a cylindrical geometry (Figure 3.1) under conditions of inhomogeneous broadening,

the emission coefficient is given by

$$E = \frac{0.90}{\tau_o(\ln\tau_o)^{0.5}} \qquad (3.52)$$

where τ_o is the optical depth at line centre[41].

3.4.2 REDUCTION OF THE RATE EQUATIONS

For the number of species considered, the relaxation matrix K_{mn} may have of the order of 10^3 components, due to the large number of energy levels n in an atom (Rydberg atoms rarely have n as large as 100), and consequently approximations must be made. However, the processes considered below are still sufficiently detailed to provide an accurate model of the processes occurring in the plasma.

(1) The One-Photon Approximation - only allows for electronic transitions between neighbouring levels, that is, those in which the principal quantum number n changes by ±1. These transitions are more frequent than those in which $\Delta n = \pm 2, \pm 3$, etc. The corresponding matrix has non-zero elements in only the principal diagonal and two adjacent levels.

(2) The probabilities of the radiative transitions from the continuous spectrum are small, and their influence on the relaxation processes in the low temperature plasma are negligible.

(3) For each species (He, He_2, Ne, Ne_2, SrI and SrII), a "pseudo-state", or lumped radiative state, has been included which is an average of higher lying levels. These states represent the most probable recombination states for the ionised species and are denoted by an asterisk in Table 3.2.

Pseudo-state	Energy above ground state (eV)
He^* (2^3P and 2^1P_1 levels)	21.1
Ne^* (^3P_o and $^3P_1$ levels)	18.5
Sr^* (5^1P and 4^1D levels)	2.6
Sr^{+*} (6^2P and 5^2D levels)	12.6
He_2^*	17.9
Ne_2^*	15.0

Table 3.2. Energy levels of the pseudo-states for each species considered in the model

The species considered in the model are listed in Table 3.3.

He (Ne)	Helium (neon) ground state	Sr	Strontium I ground state
He_m (Ne_m)	Helium (neon) metastable state	Sr^+	Strontium II ground state
He* (Ne*)	Atomic He (Ne) pseudo-state	Sr^{++}	Strontium III ground state
He^+ (Ne^+)	Singly-ionised atomic He (Ne)	Sr^*	Strontium I pseudo-state
He^{++} (Ne^{++})	Doubly-ionised atomic He (Ne)	Sr^{+*}	Strontium II pseudo-state
He_2^+ (Ne_2^+)	Singly-ionised molecular He (Ne)	Sr_2^+	$6S_{1/2}$ strontium (upper laser level)
He_2^* (Ne_2^*)	Molecular He (Ne) pseudo-state	Sr_1^+	$5P_{3/2}$ strontium (lower laser level; λ=430.5nm)
$He_2^{*'}$ ($Ne_2^{*'}$)	Unstable molecular He (Ne)	Sr_1^+,	$5P_{1/2}$ strontium (lower laser level; λ=416.2nm)
He_2^{**} (Ne_2^{**})	Excited molecular He (Ne)	Sr_m^+	$4D_{5/2}$ strontium II metastable state
$(HeNe)^+$	Molecular helium-neon ion	Sr_m^+,	$4D_{3/2}$ strontium II metastable state
$(HeNe)^*$	Unstable excited helium-neon molecule		
n_e	Free electrons		
N_υ	Photons at laser frequency υ_{21}		

Table 3.3. The atomic and optical state variables

The number of processes included in the computer model of the strontium vapour laser exceeds one hundred and thirty and these are listed together with the reaction rates/cross-sections (as appropriate) in Table B.1 in Appendix B. Figure 3.2 illustrates the processes included in the model.

3.5 THE THERMODYNAMIC EQUATIONS

The second moment solution to the Boltzmann collision equation yields the transfer rate of energy:

$$n\frac{\partial}{\partial t}\left[\frac{3kT}{2}\right] = \underline{J}.\underline{E} + \frac{3kT}{2}\underline{\nabla}_r\left[\sum_{k,i} n_{k,i}\langle\underline{v}_{k,i}\rangle\right] - n\langle\underline{v}_o\rangle.\underline{\nabla}_r\left[\frac{3kT}{2}\right] - \underline{\nabla}_r.\underline{q}$$

$$- \Pi:\underline{\nabla}_r.\langle\underline{v}_o\rangle + S_{rad} \qquad (3.53)$$

where $\underline{J}.\underline{E}$ represents the energy source and S_{rad} is the divergence of the radiation flux. This equation should be solved for all species present. An exact analytic solution is difficult without several simplifications:-

(1) The relaxation times for establishing Maxwellian distributions of all species on removal of the excitation pulse are sufficiently short. This self-equipartition time for neutrals is

$$t_s = \frac{3m}{2d^2\, n(2\pi mkT)^{0.5}} \qquad (3.54)$$

where d is the classical gas kinetic diameter. Under the conditions encountered in the SVL, these times are calculated to be of the order of 1ns and 30ns for

helium and strontium neutrals, respectively.

For charged particles of the same kind, we have[37]

$$t_s = \frac{(2m)^{0.5}\ (kT)^{1.5}}{n_2\ 4\pi\ z^4\ e_o^4 \ln\Lambda} \tag{3.55}$$

whereas for two interacting charged particle types, the relationship is considerably more complicated[18].

For electrons in dense neutral gases, the relaxation time is given by

$$t_{e\text{-}o} = \frac{3(m_e+m_a)^2}{32\ m_e\ m_a}\ \frac{1}{An_o}\left[\frac{3\pi\ m_e}{2kT_{eff}}\right] \quad . \tag{3.56}$$

(1) The relaxation time of electrons is calculated to be 5ns. Since this time is short compared with the duration of the discharge current pulse, a Maxwellian distribution of electrons is a good approximation during the afterglow.

(2) The deviation of the ionic temperatures from those of the neutrals is small[42], due to the efficiency of energy sharing between ions and neutrals of all types.

(3) A common temperature is achieved for all charged particles. The particles approach this temperature with a time constant given by

$$t_{eq}' \sim \frac{3\ m_1\ m_2\ k^{3/2}}{8(2\pi)^{1/2}\ n_2\ z_1^2 z_2^2\ e_o^4\ \ln\Lambda}\left[\frac{T_1}{m_1}+\frac{T_2}{m_2}\right]^{3/2}\left[1+\frac{n_1}{n_2}\right]^{-1/2} . \tag{3.57}$$

For strontium particles (of type 1 in equation (3.57)) in helium (of type 2), t_{eq}, is calculated to be less than 1ns. Since this time is short compared with the time scales of interest, the ion energy balance equations may be omitted. Hence, only the electron (T_e) and gas (T_g) temperatures need to be considered and these are derived from equation (3.53).

The time rate of change of the electron temperature can be written

$$\frac{d}{dt}\left[\frac{3kT_e n_e}{2}\right] = \frac{n_e e^2 E^2}{m\upsilon_m} + \sum_{ij} N_i n_e k_{ij}^s \Delta\varepsilon_{ij}^s + \sum_i N_m N_i k_p \Delta\varepsilon_i^p - \sum_{ij} N_i n_e k_{ij} \Delta\varepsilon^{ij}$$

$$- \sum_{ij} \frac{3}{2} n_e k \upsilon_{ei} (T_e - T_g)\delta_{ij} - \frac{1}{r}\nabla_r . rK(T_e)\nabla_r T_e - \frac{1}{r}\nabla_r . \left[r\frac{3}{2} n_e k T_e \langle \underline{v}_r \rangle\right] - \frac{T_e}{n_e}\frac{dn_e}{dt} \quad (3.58)$$

The first term represents the energy input from the electric field. The second term is due to superelastic collisions and the third describes Penning ionisations or atom-ion charge-transfer reactions. These terms represent the energy sources. Losses in electron energy are as follows.

The fourth term in equation (3.58) represents the electron energy loss due to inelastic collisions. The fifth term represents the electron thermalisation due to elastic collisions with atoms or ions (including Coulomb collisions); $\delta_{ij} = \frac{2m_e}{M}$ is the fraction of the electron energy given up in the process of electron thermalisation.

The next two terms represent transport processes (see Section 3.7), the first of which is the conductive heat flux, given by the Elenbaas-Heller relation[43,44] for radial and axial homogeneity. The penultimate contribution is due to radial

convection which is limited by ambipolar diffusion. Hence, the radial velocity component is

$$\langle \underline{v}_r \rangle = \mu_e E_r - D_e \frac{\underline{\nabla}_r n_e}{n_e} \tag{3.59}$$

where the radial electric field is given by

$$E_r = \frac{D_e \underline{\nabla}_r n_e + \sum_i D_1^i \sigma_1^i}{\mu_e n_e + \sum_i \mu_1^i n_1^i} \tag{3.60}$$

for ions i. Although not explicitly declared, all variables considered in the program are allowed to be temporally and radially dependent.

The analogous time rate of change of the gas temperature is:-

$$\frac{d}{dt}\left[\frac{3kT_g N_g}{2}\right] = J E_L + \sum_j \frac{3}{2} n_e k \upsilon_m^j (T_e - T_g) \delta_j + \frac{1}{r} \underline{\nabla}_r . r K_g \underline{\nabla}_r T_g \tag{3.61}$$

The second term of this equation represents coupling of the gas temperature to that of the electrons due to elastic collisions with electrons.

The local pressures of heavy particles P_{sr} and $P_{He(Ne)}$ are included in this thesis as state variables. An increase in local gas temperature increases the local pressure at the same rate. However, P_{He} relaxes to the external buffer gas pressure (P_{ext}) and P_{sr} to the vapour pressure (P_{Tw}, determined by the wall temperature) at different rates. The strontium particle pressure, P_{sr} relaxes with a time constant Λ/v_d, where Λ is the diffusion length and v_d is the speed at which strontium particles diffuse to the wall given by equation (3.59).

Similarly, P_{He} relaxes to its equilibrium value with a time constant $d/2v_s$, where v_s is the speed of sound in helium at temperature T, given by[45]

$$v_s = 972.5 \left[\frac{T}{273} \right]^{0.5} \quad . \tag{3.62}$$

This velocity is independent of gas pressure.

Finally, the local gas particle temperature T_g relaxes to the wall temperature T_w with a time constant proportional to the discharge tube radius R. Hence, the time rates of change of P_{He} and P_{Sr} are

$$\frac{dP_{He}}{dt} = \frac{\partial T_{He}}{\partial t} \frac{P_{He}}{T_{He}} - (P_{He} - P_{ext}) \frac{2v_s}{d} \tag{3.63a}$$

$$\frac{dP_{Sr}}{dt} = \frac{\partial T_{He}}{\partial t} \frac{P_{Sr}}{T_{He}} - (P_{Sr} - P_{T_w}) \frac{v_d}{R} \quad . \tag{3.63b}$$

Assuming the gas density stays in equilibrium with the local temperature and pressure, then the equations describing the time rates of change of helium and strontium neutral concentrations should also include a contribution due to changes in temperature and pressure of value

$$\frac{dN_{He}}{dt} = - \frac{N_{He}}{T_{He}} \frac{\partial T_{He}}{\partial t} + \frac{N_{He}}{P_{He}} \frac{\partial P_{He}}{\partial t} = - (P_{He}-P_{ext}) \frac{N_{He}\, 2v_s}{d\, P_{He}} \tag{3.64a}$$

$$\frac{dN_{Sr}}{dt} = - \frac{N_{Sr}}{T_{He}} \frac{\partial T_{He}}{\partial t} + \frac{N_{Sr}}{P_{Sr}} \frac{\partial P_{Sr}}{\partial t} = - (P_{Sr}-P_{T_w}) \frac{N_{Sr}\, v_d}{\Lambda\, P_{Sr}} \quad . \tag{3.64b}$$

The thermodynamic state variables are listed in Table 3.4.

T_e	Free electron temperature	$P_{He(Ne)}$	Helium (neon) gas pressure
T_g	Helium (neon) or strontium temperature	P_{Sr}	Strontium vapour pressure

Table 3.4. The thermodynamic state variables

3.6 THE CIRCUIT EQUATIONS

A detailed circuit model is included in the computer model. The basic modulator circuit is shown in Figure 3.3 and consists of a high-voltage power supply which resonantly charges the storage capacitance C_s via a charging inductor L_c and a bypass inductor L_b. When the thyratron is triggered, the anode voltage drops to the arc drop level and C_s starts to discharge. If the value of L_b is high enough, only an insignificant amount of the discharge current flows through it. The voltage across the laser increases until the gas breaks down, at which point, C_s discharges into the laser.

Although straightforward in appearance, this modulator circuit contains a voltage-dependent impedence (the laser load), a time-dependent switch element (the thyratron) and, possibly, a current-dependent inductance (the saturable bypass inductor - Chapter Six). A lumped parameter representation of the circuit containing user-defined non-linear elements is used to simulate the modulator and is shown in Figure 3.4.

The laser plasma is represented by a series-connection of a resistance R_d and an inductance L_d. The former is given by

$$R_d = \frac{d}{\sigma_d A} \tag{3.65}$$

where A and d are, respectively, the discharge cross-sectional area and length. Combining this with equation (3.86) for electrical conductivity σ_d, this gives

$$R_d = \frac{3\ k\ T_e\ d}{n_e\ e^2 <\underline{v}_e>\ A} \sum_i N_i\ \sigma_{mi} \tag{3.66}$$

The inductance can be written

$$L_d = \frac{m_e d}{n_e e^2 A} \tag{3.67}$$

Computer modelling[46] has revealed that the thyratron can be represented by an ideal switch in series with a voltage-dependent resistance and a constant inductance L (Figure 3.5). Prior to triggering, the thyratron resistance is of the order of 100kΩ. In laser circuits which produce short pulses, the value of this resistance after a time t_{ON} (to represent the finite thyratron turn-on time) has dropped to a breakdown value of, typically, 0.1 ohms, dependent on the di/dt and current reversal. An important result of this work is the demonstration that the rate of rise of anode current is limited predominantly by circuit inductance and not by plasma development. The turn-on time t_{ON} is determined predominantly by the tube gas pressure and, to a lesser extent, by the initial anode voltage and circuit parameters; for circuits of the type considered in this thesis, t_{ON} is typically 10ns. A series diode is included in the equivalent circuit of the unidirectional thyratron to prevent reverse conduction.

Most of the current return paths in the rest of the circuit are coaxial, simplifying inductance calculations. The inductance of the storage capacitor bank is of the order of a few hundred picohenries[47] and that of the current-viewing resistors is a few nanohenries[48]; both are negligible compared with the inductances of the other circuit elements.

The relevant state variables are listed in Table 3.5.

V_{cs}	Storage capacitor voltage	i_{Lt}	Thyratron current
V_{cl}	Smoothing capacitor voltage	i_{Ll}	Laser current
		i_{Lb}	Bypass inductor current
		i_{Lc}	Charging inductor current

Table 3.5. The electrical state variables

3.7 TRANSPORT PROCESSES

Diffusion and electrical conductivity are important considerations in the zero-order moment solutions of the Boltzmann collision equation. Expressions for thermal conductivity are obtained from second order solutions.

The transport coefficients a^s associated with a property ϕ^s can be written

$$<\phi^{s+1}> = - a^s \nabla_{\underline{r}} <\phi^{s+1}> = - a^s \nabla_{\underline{r}} <\phi^s w> \qquad (3.68)$$

and are obtained analytically as velocity moments of the Boltzmann equation.

As mentioned earlier, the degree of ionisation of a decaying plasma and its

properties are a function of time. In a pulsed gas discharge, it is necessary to consider a plasma of arbitrary ionisation. This allows a description of the time rate of change of all variables which must be integrated over one complete discharge cycle to achieve a picture of the kinetics of the plasma build-up and decay.

3.7.1 DIFFUSION

The particle flux density ϕ is given by

$$\phi = - D \nabla_r n - D k^T \frac{\nabla_r T}{T} \tag{3.69}$$

where D is the diffusion coefficient and k^T is the thermodiffusion factor (=n/2[18]; important if strong temperature gradients are present).

Diffusion of neutrals is treated as a binary process of particles through the neutral (buffer) gas. For overall charge neutrality in a multicomponent mixture[17,18], solution of the two-fluid model for electrons and ions gives the diffusion coefficient of the i^{th} particle as

$$D = \frac{9\pi}{16} \frac{1}{3} \langle \lambda_i \rangle \langle v_i \rangle \tag{3.70}$$

where the mean free path is found analytically[18] to be

$$\langle \lambda_i \rangle = \frac{1}{2^{0.5} n_i \sigma_i^j + n_j \sigma_i^j \left[1 + \frac{m_i}{m_j}\right]} \tag{3.71}$$

where σ_i^j is the cross-section for encounters between particles of types i and j

and $\langle v_i \rangle$ is the average velocity of the particles involved:

$$\langle v_i \rangle = \left[\frac{8\,k\,T}{\pi m} \right]^{0.5} \quad . \tag{3.72}$$

Then, the individual fluxes are

$$\phi_i = - D \frac{\sum n}{n_i} \nabla_r \left[\frac{n_i}{\sum n} \right] \quad . \tag{3.73}$$

When space-charge forces dominate, the ambipolar diffusion coefficient for charged particles is given by

$$D_a = D_+ (1 + T_e/T_i) \quad . \tag{3.74}$$

In a multi-positive-ion ($He^+(Ne^+)$, $He^{++}(Ne^{++})$, $He_2^+(Ne_2^+)$, Sr^+, Sr^{++}) discharge, the ambipolar diffusion coefficient is modified to

$$D_{a,e} \approx \frac{\sum_i n}{n_e} D_{a,i} \tag{3.75a}$$

$$D_{a,i} \approx D_i (1 + T_e/T_i) \quad . \tag{3.75b}$$

However, for an arbitrary degree of ionisation, the diffusion process is intermediate between free and ambipolar. In general, the effective coefficient for charged particles is[49]

$$D_s = \frac{D_i + \Lambda^2 \sigma_c/\varepsilon_o}{D_a + \Lambda^2 \sigma_c/\varepsilon_o} D_a \tag{3.76}$$

where σ_c is the conductivity at the centre of the discharge:

$$\sigma_c = (\mu_+ n_+ + \mu_e n_e) \tag{3.77}$$

and Λ is the characteristic diffusion length which, for cylindrical symmetry, is[31]

$$\frac{1}{\Lambda^2} = \left[\frac{\pi}{d}\right]^2 + \left[\frac{2.405}{R}\right]^2 \quad . \tag{3.78}$$

3.7.2 ELECTRICAL CONDUCTIVITY

The current density in a plasma is given by

$$j = \sigma \nabla V \quad . \tag{3.79}$$

The electronic contribution to the electrical conductivity consists of the ordinary transport of charges j_e due to the radial ambipolar electric field together with a convective contribution, j_{th}. This thermocurrent is due to the thermal gradients.

The total current density is

$$\begin{aligned} j &= j_e + j_{th} \\ &= \sigma_e E_r + \tfrac{1}{3} e \lambda_e \nabla_r (n_e \langle v_e \rangle) \quad . \end{aligned} \tag{3.80}$$

The electrical conductivity σ_e is given by

$$\sigma_e = e\, n_e\, \mu_e \tag{3.81}$$

where μ_e is the electron mobility.

When the degree of ionisation is greater than approximately 1%, σ_e is a function not only of electron-neutral collisions, but also of electron-ion and electron-electron encounters[50]. Hence, we have

$$\mu_e = \frac{e}{m_e} \frac{1}{\upsilon_e^{total}} \tag{3.82}$$

where the total electron collision frequency is given by

$$\upsilon_e^{total} = \sum_i n_i \langle \sigma_e^i\, v_e \rangle + \sum_a n_a \langle \sigma_e^a\, v_e \rangle \quad . \tag{3.83}$$

The term σ_e^i is the Coulomb cross-section and σ_e^a is the cross-section for electron-atom collisions. The two components of equation (3.83) are derived from[18]

$$\upsilon_{ei} = \frac{(8\pi)^{0.5}}{3} z_i^2 \frac{e_o^4\, n_i\, \ln\Lambda}{m\,(kT_e)^{1.5}} \tag{3.84a}$$

and

$$\upsilon_{ii} = \frac{2^{1.5}}{5} \left[\frac{m_e}{m_i}\right]^{0.5} \upsilon_{ei} \tag{3.84b}$$

where

$$\ln\Lambda \approx \ln\left[\frac{4kT\rho_D^2}{e_o^2}\right] \tag{3.85}$$

in which ρ_D is the Debye length.

Substituting for υ (given by equations (3.82) and (3.83)) into equation (3.81) gives

$$\sigma = 3.05 \times 10^{-10} \frac{n_e}{T_e^{0.5}} \frac{1}{\sum_i n_i Q_i} \quad . \tag{3.86}$$

The thermocurrent is given by[18]

$$j_{th} = \frac{1}{3} e \lambda_e \nabla_r (n_e \langle v_e \rangle) \quad . \tag{3.87}$$

Combining equations (3.72) and (3.87) gives

$$j_{th} = \frac{1}{3} \left[\frac{8k}{\pi m_e} \right]^{0.5} e \lambda_e \nabla_r (n_e T_e^{0.5}) \quad . \tag{3.88}$$

3.7.3 THERMAL CONDUCTIVITY

The heat flux density of a plasma is

$$q = - K \nabla_r T - K T \frac{\nabla_r n}{n} \tag{3.89}$$

where K is the normal coefficient of contact thermal conductivity. The second term is the heat flux density due to thermodiffusion, q_{th}. Equation (3.89) can be written in the form

$$q = - K \left[1 + \frac{T}{n} \nabla T^n \right] \nabla_r T$$

$$= - K_{TOT} \nabla_r T \quad . \tag{3.90}$$

For a weakly-ionised plasma, the thermal conductivity is dominated by the neutral particles; reference 51 gives

$$K (T_g) \approx \frac{(T_g/M_g)^{0.5}}{d^2 \Omega} \tag{3.91}$$

where d^2 is the characteristic molecular dimension of the gas concerned and Ω is the collision integral for thermal conductivity.

At high degrees of ionisation (discharge currents in excess of 50A), the electronic contribution dominates due to the high thermal velocities $\langle v_e \rangle$ of electrons. For a fully-ionised (Lorentz) gas, Spitzer[37] gives

$$K \approx \frac{T^{2.5}}{z \ln\Lambda} \tag{3.92}$$

where $\ln\Lambda \alpha \left(\frac{T^3}{n_e} \right)^{0.5}$.

For an arbitrary degree of ionisation, both Hirschfelder[51] and Cohen[52] consider a weighted average of the electronic and neutral contributions to the thermal conductivity. In the latter case, we have

$$\frac{1}{K_{TOT}} = \frac{(1 - f)}{K_i (T/T_i)} + \frac{f}{K_F} \tag{3.93}$$

where f is the degree of ionisation of the gas ($=n_e/N_{He}$),

K_i is the thermal conductivity at the ionisation temperature

and K_F is the Spitzer value for a fully-ionised gas.

Here, a more general description is necessary. The total thermal conductivity of the plasma is the sum of the contributions from each species i present, that is

$$K_{TOT} = \sum_i K_i$$

$$= \sum_m K_{m'} + \sum_n (K_{n'} + K_n^{react}) + \sum_i (K_{i'} + K_i^{react}) + K_e + K_{Th,D} \quad (3.94)$$

Molecules Atoms Ions Electrons Thermodiffusion

The coefficients of contact thermal conductivity are given by

$$K_{a'} = \frac{1}{3} \lambda_a \langle v_a \rangle n C_v^a \quad (3.95)$$

where C_v^a is the specific heat at constant volume for species a. At low temperatures ($<5000^oK$), $C_v^a = \frac{3}{2}k$. This ranges up to $(z+1)\frac{3}{2}k$ for a completely ionised particle of atomic number z.

The terms K_n^{react} and K_i^{react} represent the reaction energies present in the processes of dissociative recombination and ionisation-recombination, respectively, given by

$$K_n^{react} = \frac{1}{n_n} \frac{D_{o,a}}{2} \frac{dn_n}{dt} \quad (3.96a)$$

$$K_i^{react} = \frac{2}{m_i} \frac{E_i}{n_n} \frac{dn_i}{dt} \quad (3.96b)$$

where $D_{o,a}$ is the dissociation and E_i the ionisation energy. If neutral atoms are produced by dissociation, then each atom transports half of the dissociation energy.

The term K_D is the thermal transport due to simple diffusion. It is given by

$$K_{D,i} = \frac{15\pi}{8}^{0.5} k \left[\frac{kT}{m_a}\right]^{0.5} \left[1+\frac{E_i}{5kT/2}\right] \frac{1}{\langle\sigma_a^i\rangle} \left[\frac{1}{1+n_a/n_e}\right] \left[1+\frac{T_e}{n_e}\frac{dn_e}{dT}\right] \quad (3.97)$$

Finally, substituting equations (3.95), (3.96) and (3.97) into equation (3.94), we get

$$K_{TOT} = \frac{1}{3}\lambda_m\langle v_m\rangle n_m C_v^m + \frac{1}{3}\lambda_a\langle v_a\rangle n_a C_v^a \left[1+\frac{T_a}{n_a}\frac{dn_a}{dT}\right] + \frac{1}{n_a}\frac{D_{o,a}}{2}\frac{dn_a}{dt}$$

$$+ \frac{1}{3}\lambda_i\langle v_i\rangle n_i C_v^i \left[1+\frac{T_i}{n_i}\frac{dn_i}{dT}\right] + \frac{2}{n_i}\frac{E_i}{m_i}\frac{dn_i}{dt} + \frac{1}{3}\lambda_e\langle v_e\rangle k\frac{n_e}{2} \quad . (3.98)$$

3.8 STATE VARIABLE DESCRIPTION OF THE LASER CIRCUIT AND LEVEL POPULATIONS WITH VARIABLE COEFFICIENTS

The formal definition of the state of a dynamic system is "a set of physical quantities which (in the absence of external excitation) completely determines the evolution of the system". State variables describe the internal states of the pulsed gas discharge laser system. There may exist m atomic (number density of particles in a particular state, free electron density), n thermodynamic (local particle temperatures and pressures, energy densities of the gas and electrons), p electrical (capacitance charges and inductance fluxes) and q optical (number of photons with a particular energy) state variables. These are the independent initial conditions which the system can support. If the state variables are designated N^m, T^n, E^p and P^q respectively, then the system can be described by coupled rate equations describing the time evolution of these quantities. In the state-space model, the differential equations take the form:

$$
\begin{aligned}
\dot{N}_1 &= a_{1,1}N_1 + \ldots + a_{1,m}N_m + a_{1,m+1}T_1 + \ldots + a_{1,m+n}T_n \\
&\quad + a_{1,m+n+1}E_1 + \ldots + a_{1,m+n+p}E_p + a_{1,m+n+p+1}P_1 + \ldots + a_{1,m+n+p+q}P_q \\
&\vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \\
\dot{N}_m &= a_{m,1}N_1 + \ldots + a_{m,m}N_m + a_{m,m+1}T_1 + \ldots + a_{m,m+n}T_n \\
&\quad + a_{m,m+n+1}E_1 + \ldots + a_{m,m+n+p}E_p + a_{m,m+n+p+1}P_1 + \ldots + a_{m,m+n+p+q}P_q \\
\dot{T}_1 &= a_{m+1,1}N_1 + \ldots + a_{m+1,m}N_m + a_{m+1,m+1}T_1 + \ldots + a_{m+1,m+n}T_n \\
&\quad + a_{m+1,m+n+1}E_1 + \ldots + a_{m+1,m+n+p}E_p + a_{m+1,m+n+p+1}P_1 + \ldots + a_{m+1,m+n+p+q}P_q \\
&\vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \\
\dot{T}_n &= a_{m+n,1}N_1 + \ldots + a_{m+n,m}N_m + a_{m+n,m+1}T_1 + \ldots + a_{m+n,m+n}T_n \\
&\quad + a_{m+n,m+n+1}E_1 + \ldots + a_{m+n,m+n+p}E_p + a_{m+n,m+n+p+1}P_1 + \ldots + a_{m+n,m+n+p+q}P_q \\
\dot{E}_1 &= a_{m+n+1,1}N_1 + \ldots + a_{m+n+1,m}N_m + a_{m+n+1,m+1}T_1 + \ldots + a_{m+n+1,m+n}T_n \\
&\quad + a_{m+n+1,m+n+1}E_1 + \ldots + a_{m+n+1,m+n+p}E_p \\
&\quad + a_{m+n+1,m+n+p+1}P_1 + \ldots + a_{m+n+1,m+n+p+q}P_q \\
&\vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \\
\dot{E}_p &= a_{m+n+p,1}N_1 + \ldots + a_{m+n+p,m}N_m + a_{m+n+p,m+1}T_1 + \ldots + a_{m+n+p,m+n}T_n \\
&\quad + a_{m+n+p,m+n+1}E_1 + \ldots + a_{m+n+p,m+n+p}E_p \\
&\quad + a_{m+n+p,m+n+p+1}P_1 + \ldots + a_{m+n+p,m+n+p+q}P_q \\
\dot{P}_1 &= a_{m+n+p+1,1}N_1 + \ldots + a_{m+n+p+1,m}N_m + a_{m+n+p+1,m+1}T_1 + \ldots + a_{m+n+p+1,m+n}T_n \\
&\quad + a_{m+n+p+1,m+n+1}E_1 + \ldots + a_{m+n+p+1,m+n+p}E_p \\
&\quad + a_{m+n+p+1,m+n+p+1}P_1 + \ldots + a_{m+n+p+1,m+n+p+q}P_q \\
&\vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \\
\dot{P}_q &= a_{m+n+p+q,1}N_1 + \ldots + a_{m+n+p+q,m}N_m + a_{m+n+p+q,m+1}T_1 + \ldots + a_{m+n+p+q,m+n}T_n \\
&\quad + a_{m+n+p+q,m+n+1}E_1 + \ldots + a_{m+n+p+q,m+n+p}E_p \\
&\quad + a_{m+n+p+q,m+n+p+1}P_1 + \ldots + a_{m+n+p+q,m+n+p+q}P_q
\end{aligned}
\tag{3.99}
$$

In vector notation, we have

$$\dot{x}_k(\underline{r},t) = \frac{dx_k(\underline{r},t)}{dt} = A(\underline{r},t)x_k(\underline{r},t) + B(\underline{r},t)u_k(\underline{r},t) \qquad (3.100)$$

where u is the external input vector. The number of state variables $x_k(\underline{r},t)$ equals the order of the system and these are listed in Tables 3.3, 3.4 and 3.5. The output vector is given by

$$y_k(\underline{r},t) = C(\underline{r},t)x_k(\underline{r},t) + D(\underline{r},t)u_k(\underline{r},t) \quad . \qquad (3.101)$$

For this system, A, B, C and D are spatially and temporally non-linear, four-dimensional matrices.

REFERENCES FOR CHAPTER 3

1. R. Künnemeyer, C. W. McLucas, D. J. W. Brown and A. I. McIntosh, "Time-resolved measurements of population densities in a Sr^+ recombination laser", IEEE J. Quantum Electron. QE-23 (11), p.2028 (1987).

2. D. R. Bates, A. E. Kingston and R. W. P. McWhirter, "Recombination between electrons and atomic ions. I. Optically thin plasmas", Proc. Roy. Soc., A267, p.297 (1962).

3. L. I. Gudzenko and L. A. Shelepin, "Radiative enhancement in a recombining plasma", Sov. Phys. - Dokl., 10, No.2, p.147 (1965).

4. L. I. Gudzenko, A. T. Mamachun and L. A. Shelepin, "Hydrogen level populations in a pulsed recombination plasma", Sov. Phys. - Tech. Phys., 12, No.5, p.598 (1967).

5. B. F. Gordiets, L. I. Gudzenko and L. A. Shelepin, "Relaxation processes and amplification of radiation in a dense plasma", Sov. Phys. - JETP., 28, No.3, p.489 (1969).

6. V. V. Zhukov, V. S. Kucherov, E. L. Latush and M. F. Sém, "Recombination lasers utilizing vapours of chemical elements. II. Laser action due to transitions in metal ions", Sov. J. Quantum Electron., 7, No.6, p.708 (1977).

7. V. E. Prokop'ev and V. I. Solomonov, "Investigation of a strontium vapour laser", Sov. J. Quantum Electron. 15 (6), p.832 (1985).

8. R. J. Carman, "A self-consistent model for a longitudinal discharge excited He-Sr recombination laser", IEEE J. Quantum Electron., QE-26 (9), p.1588 (1990).

9. M. J. Kushner, "A self-consistent model for high repetition rate copper vapour lasers", IEEE J. Quantum Electron., QE-17 (8), p.1555 (1981).

10. M. J. Kushner and B. E. Warner, "Large-bore copper vapour lasers: Kinetics and scaling issues", J. Appl. Phys., 54 (6), p.2970 (1983).

11. M. J. Kushner and F. E. C. Culick, "A model for the dissociation pulse, afterglow and laser pulse in the Cu/CuCl double pulse laser", J. Appl. Phys., 51, No.6, p.3020 (1980).

12. W. Schottky, Phys. Z., 25, p.342 (1924).

13. W. Schottky, Phys. Z., 24, p.635 (1924).

14. G. L. Rogoff, "Gas heating effects in the constriction of a high-pressure glow discharge column", Phys. Fluids, 15, No.11, p.1931 (1972).

15. E. F. Jaeger, L. Oster and A. V. Phelps, "Growth of thermal constrictions in a weakly ionized gas discharge in helium", Phys. Fluids, 19, No.6, p.819 (1976).

16. G. Ecker and O. Zoller, "Thermally inhomogeneous plasma column", Phys. Fluids, 7, No.12, p.1996 (1964).

17. J. G. Eden and B. E. Cherrington, "Radial neutral gas temperature and density profiles in low-pressure argon discharges", J. Appl. Phys., 44 (11), p.4920 (1973).

18. H. W. Drawin, "Non-equilibrium plasmas", in 'Reactions Under Plasma Conditions: Volume 1', ed. M. Venugopalan, Wiley Interscience, Ch.3, Section IV, pp.125-238 (1971).

19. L. I. Gudzenko, L. A. Shelepin and S. I. Yakovlenko, "Amplification in recombining plasmas (plasma lasers)", Sov. Phys. - Usp., 17, No.6, p.848 (1975).

20. D. R. Bates and A. Damgaard, "The calculation of the absolute strengths of spectral lines", Phil. Trans. Roy. Soc., A242, p.101 (1949).

21. L. Goldberg, "Relative multiplet strengths in LS coupling", Astrophys. J., 82, p.1 (1935).

22. L. Goldberg, "Note on absolute multiplet strengths", Astrophys. J., 84, p.11 (1936).

23. Von R. de L. Krönig, "Über die Intensität der Mehrfachlinien und ihrer Zeemankomponenten. II", Zs. f. Phys., 33, p.261 (1925).

24. H. N. Russell, "Tables for intensities of lines in multiplets", Astrophys. J., 83, p.129 (1936).

25. H. E. White and A. Y. Eliason, "Relative intensity tables for spectrum lines", Phys. Rev., 44, p.753 (1933).

26. I. I. Sobel'man, 'Introduction to the Theory of Atomic Spectra', First edition, Pergamon Press, p.338 (1972).

27. R. F. Bacher and S. Goudsmit, 'Atomic Energy States', First edition, McGraw-Hill Book Co. (1932).

28. A. Gallagher, "Oscillator strengths of CaII, SrII and BaII", Phys. Rev., 151, No.1, p.24 (1967).

29. I. I. Sobel'man, 'Introduction to the Theory of Atomic Spectra', First edition, Pergamon Press (1972).

30. W. L. Fite, "The measurement of collisional excitation and ionisation cross sections", in 'Atomic and Molecular Processes', ed. D. R. Bates, Academic Press, pp.421-492 (1962).

31. B. E. Cherrington, "Distribution functions and the Boltzmann equation", in 'Gaseous Electronics and Gas Lasers', First edition, Pergamon Press, Ch.4, pp.53-57 (1980).

32. E. C. G. Stuckelberg and P. M. Morse, "Computation of the effective cross section for the recombination of electrons with hydrogen ions", Phys. Rev., 36, p.16 (1930).

33. I. S. Veselovskii, "Electron recombination coefficient with three-body collisions in a plasma", Sov. Phys. - Tech. Phys., 14, No.2, p.193 (1969).

34. N. D'Angelo, "Recombination of ions and electrons", Phys. Rev., 121, No.2, p.505 (1961).

35. L. M. Biberman, V. S. Vorob'ev and I. T. Yakubov, "Kinetics of impact-radiative ionisation and recombination", Sov. Phys. - Usp., 15, No.4, p.375 (1973).

36. B. Makin and J. C. Keck, "Variational theory of three-body electron-ion recombination rates", Phys. Rev. Lett., 11, p.281 (1963).

37. L. Spitzer, 'Physics of Fully Ionized Gases', Interscience, New York (1962).

38. E. Hinnov and J. G. Hirschberg, "Electron-ion recombination in dense plasmas", Phys. Rev., 125, No.3, p.795 (1962).

39. D. R. Bates, "Dissociative recombination", Phys. Rev., 78, p.492 (1950).

40. I. Ya. Fugol, P. L. Pakhomov and G. P. Reznikov, "Spectroscopic investigation of a pulsed high-frequency discharge in helium", Optics Spectrosc., 16, p.510 (1964).

41. T. Holstein, "Imprisonment of resonance radiation in gases", Phys. Rev., 72, No.12, p.1212 (1947).

42. L. Goldstein and T. Sekiguchi, "Electron-electron interaction and heat conduction in gaseous plasmas", Phys. Rev., 109, No.3, p.625 (1958).

43. W. Elenbaas, "Die Temperatür des Quecksilberbogens", Physica (Utr.), 1, p.211 (1934).

44. W. Elenbaas, "Die Quecksilber-hochdrückentladung", Physica (Utr.), 1, p.673 (1934).

45. G. W. C. Kaye and T. H. Laby, 'Tables of Physical and Chemical Constants', Fifteenth edition, Longman, London and New York, p.71 (1986).

46. H. Menown, C. A. Pirrie and N. S. Nicholls, "Advanced thyratrons as switches for the nineties", IEEE Seventeenth Power Modulator Symposium, Seattle, WA, pp.69-73, June, 1986.

47. Murata Mfg. Co., Ltd., Nagaokakyo-shi, Kyoto 617, Japan.

48. T&M Research Products, Inc., 'Series W - Current Viewing Resistors Products Data', p.11, Albuquerque, NM (6/1/81).

49. W. P. Allis and D. J. Rose, "The transition from free to ambipolar diffusion", Phys. Rev., 93, No.1, pp.84-93 (1954).

50. S. Chapman and T. G. Cowling, 'The Mathematical Theory of Non-uniform Gases', Second edition, Cambridge University Press, Cambridge (1970).

51. J. O. Hirschfelder, C. F. Curtiss and R. B. Bird, 'Molecular Theory of Gases and Liquids', Wiley, New York (1954).

52. I. M. Cohen and A. M. Whitman, "Unified positive column theory of gas discharges", Phys. Fluids, 16, No.2, p.307 (1973).

FIGURES FOR CHAPTER 3

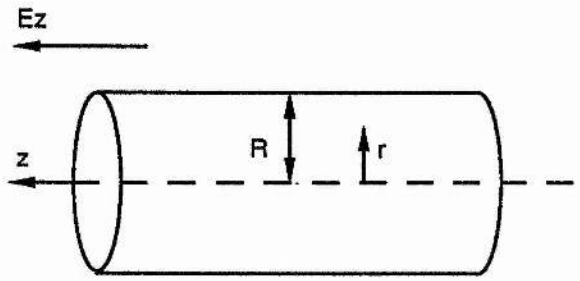


Figure 3.1. Cylindrical geometry used in the model

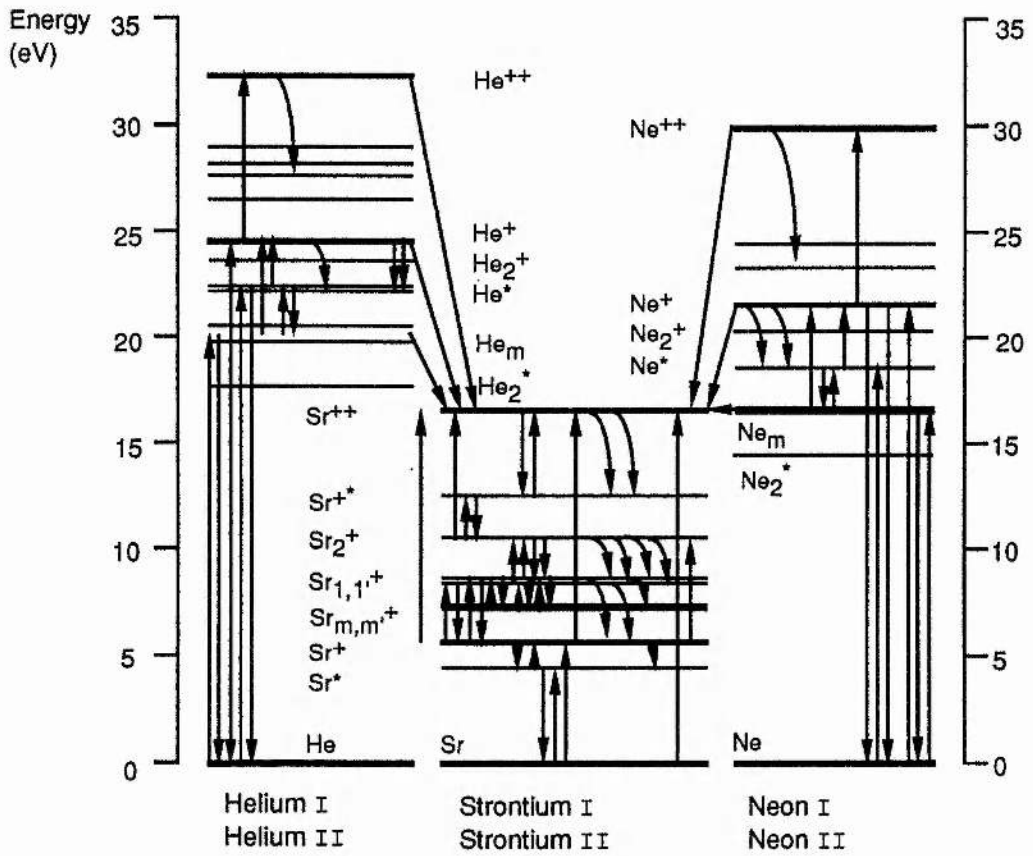


Figure 3.2. A schematic representation of collisional and radiative processes included in the model

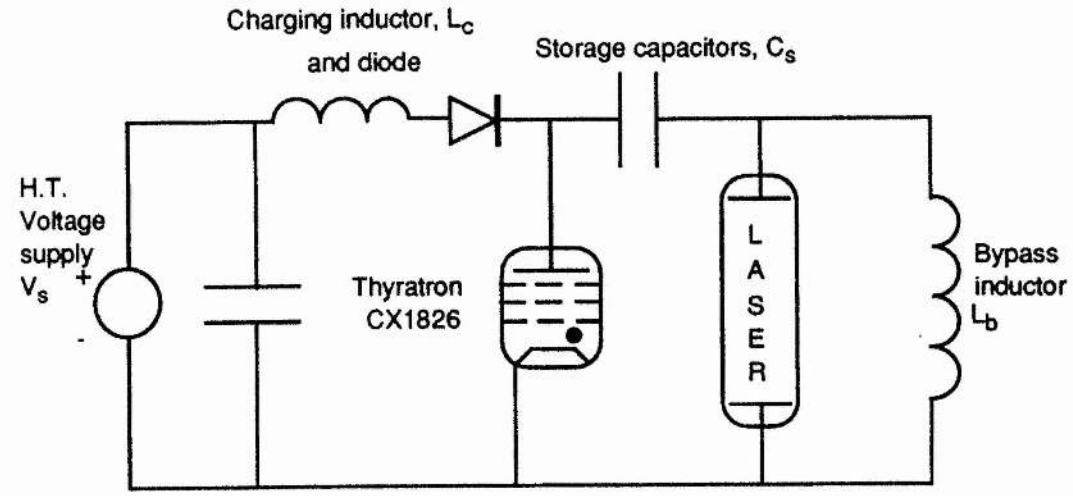


Figure 3.3. Ideal laser modulator

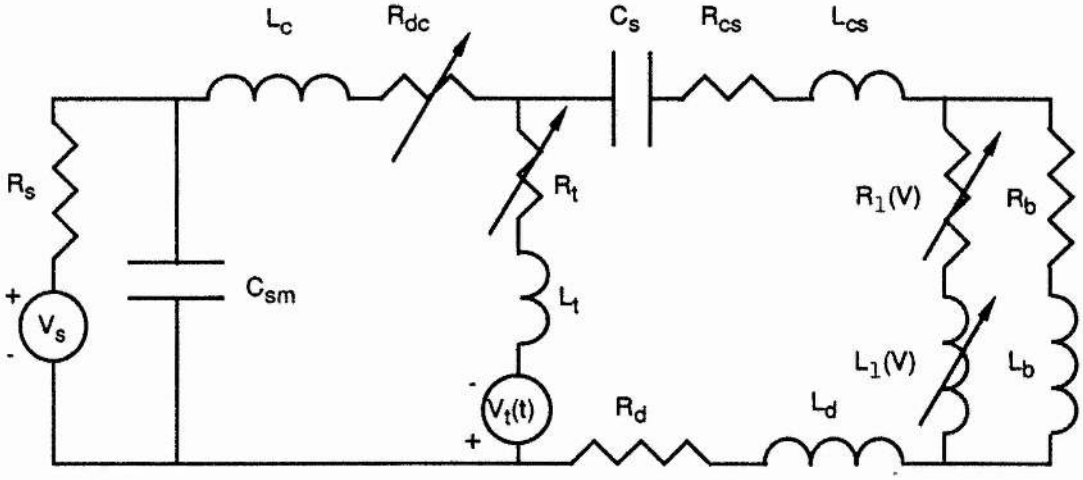


Figure 3.4. Practical circuit used to simulate the laser modulator

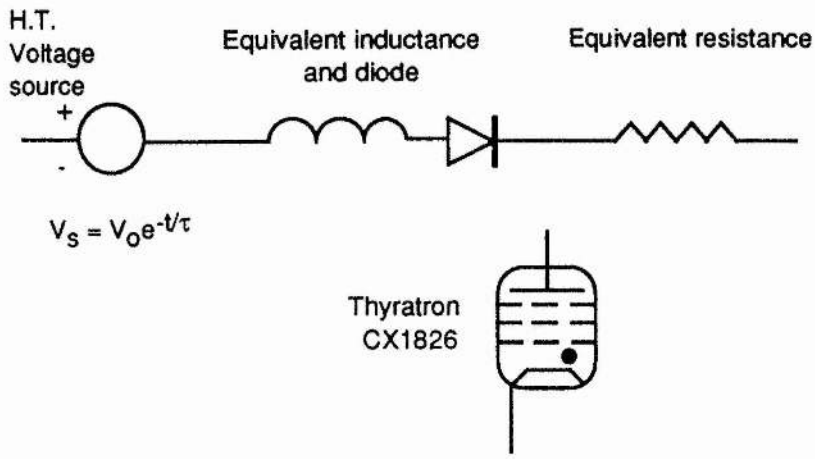


Figure 3.5. Equivalent circuit for a unidirectional thyratron

Chapter 4

Computer Model of Rate Processes in the Strontium Vapour Laser

4.1 INTRODUCTION

The concepts outlined in Chapter Three are brought together to form the basis of a computer model which simulates a pulsed electrical discharge in a helium buffer gas/strontium vapour (and possibly neon) mixture. The aim of the model is to provide a description of the complex processes occuring within the plasma both during the discharge pulse and in the afterglow period. The afterglow domain is particularly important for recombination processes and determines the initial conditions (electron density) for the next pumping pulse. A state-variable description forms the basis of a homogeneous collisional-radiative model in the zero- and second-order momentum approximation. This kinetic model, known as SRRAD, is applicable to a longitudinal electrode configuration and is used to examine the effects of parametric variation of the driving circuit on the development of microscopic parameters such as species densities and small-signal gain.

The model may be used for laser system design. The accuracy of the simulations can only be checked by testing the ability of the model to reproduce specific experimental results. Experimental techniques can be broadly classified into two categories:

(1) macroscopic parameters such as voltage, current and laser output

power

and (2) spectra of transient excited species.

The numerical results obtained regarding the dependence of laser intensity on capacitor charging voltage, storage capacitance and pulse repetition frequency of the excitation circuit are compared with experimental observations in Chapter Five. Spectroscopic information regarding transient population densities of excited/ionic states in the strontium laser has recently been reported[1], and, where appropriate, the results of the model are compared with this information.

The discharge is assumed to be homogeneous. This "zero-dimensional" model is believed to be reasonably accurate for predicting the performance of small-bore SVLs up to approximately 10-15 mm bore diameter. For larger diameter discharge tubes, radial temperature gradients lead to significant variations in species densities across the tube cross-section which tends to increase beam non-uniformity. For this reason, all state variables in the model are allowed to be radially dependent which will allow an inhomogeneous development to the model if necessary.

4.2 NUMERICAL ANALYSIS

The set of coupled first order differential equations (3.6), (3.53), (3.63) and (3.65), together with the general expression for transport processes (3.68) cast in state variable form (equations (3.100) and (3.101)) completely describe the kinetics of the SVL. The reaction rates and cross-sections of the chemical reactions included in the model are listed in Table B.1 in Appendix B. Twenty six chemical species, including electrons and excited states as separate species, are included. The rates of electron impact excitation are calculated

by convoluting the cross-section for the process with a Maxwellian electron velocity distribution appropriate to the current electron temperature (equation (3.27)). Values of oscillator strengths for the relevant transitions are obtained from Table 3.1. The low energy (< 10eV) excitation rates for the helium, neon and strontium transitions used in the model are illustrated as a function of electron temperature in Figure 4.1. As will be shown in this chapter, electron-impact processes dominate over stimulated emission in the determination of the populations of laser levels. For this reason, only a simple optical model is included.

4.2.1 BOUNDARY CONDITIONS

In the centre of the plasma, the populations and temperatures have no singularities, so we have

$$r = 0 : \frac{\partial N}{\partial r} = \frac{\partial T_n}{\partial r} = \frac{\partial n_e}{\partial r} = 0 \quad . \tag{4.1}$$

Also, since the neutral temperature and density and the charged particle density is assumed to be azimuthally uniform, there is no radial flow at the centre, that is, we have

$$r = 0 : V_N = 0 \quad . \tag{4.2}$$

On the other hand, at the edge of the discharge column, the charged particle density is zero. Further, the neutral temperature and pressure at the wall are taken to be those of the wall T_w and the external buffer gas pressure P_{ext}, respectively, so we write

$$r = R : T_N = T_w,\ P = P_{ext},\ N = \text{finite},\ n_e = 0 \quad . \qquad (4.3)$$

Initially, N, T and V_N are spatially uniform and of value N_o, T_o and zero, respectively. An initial electron density is assumed with a spatial distribution consistent with a diffusion controlled growth of ionisation:-

$$t = 0 : n_e(r,0) = n_e(0,0)\, J_o\,(2.405\ r/R) \quad cm^{-3} \quad , \qquad (4.4)$$

where J_o is a Bessel function of the first kind and order zero[2]. The value of $n_e(0,0)$ is assumed. Since the gas density is uniform at this time, the electron drift velocity is independent of radius. Hence, the current density profile is described by the electron density distribution (4.4). Combining equations (3.82), (3.83) and (4.4), we get

$$j_e(r,0) = \frac{n_e(r,0) e \mu_e}{A} = \frac{e^2\, n_e(0,0)\ J_o(2.405 r/R)}{m_e\, A \sum_i n_i \langle \sigma_e v_e \rangle} \quad . \qquad (4.5)$$

4.2.2 INITIAL CONDITIONS

At time t = 0, only electrons and singly-ionised strontium ions together with ground state helium, neon and strontium I atoms are assumed to be present. The electrons exhibit a Bessel function distribution (equation (4.4)). All temperatures and pressures are assumed to be in equilibrium. The former are equal to the wall temperature; helium and neon particles are at the external buffer gas pressure whilst strontium particles exhibit the vapour pressure corresponding to the wall temperature[3]. These assumptions are valid for an operating frequency in the region of 1 kHz, which corresponds to an interpulse period of 1 ms. According to equations (3.62) and (3.63a), the relaxation time

of the helium pressure to the external buffer gas pressure under these conditions is of the order of 200-300μs.

4.2.3 TEMPORAL DEVELOPMENT

The excitation circuit is shown in Figure 3.3. The storage capacitance C_s is resonantly charged to twice the power supply voltage. Once the thyratron is switched, the voltage across the discharge tube increases, accompanied by a rise in electron temperature, until the gas breaks down. The heated electrons collide with helium and strontium atoms, creating the desired population inversion and ultimately heating the gas in the discharge. Subsequent cooling of the gas by conduction to the walls heats the discharge tube to the correct temperature to maintain the optimum vapour pressure for lasing. The variation in electron temperature due to the various inelastic scattering processes such as ionisation and recombination is recorded. Equations describing the time-dependence of species densities (3.6), electron and gas temperatures (3.58, 3.61), discharge voltage and current and photon density (3.50) are simultaneously integrated by means of the method of finite differences. The form of the rate equations which describe these processes is given in the program listing of the computer model (Appendix D).

The state variables $x_n(\underset{\sim}{r},t)$ are evaluated at a time $t+\Delta t$ by means of the method of finite differences applied to the set of initial profiles as initial conditions. Once the plasma description at all points $x_n(\underset{\sim}{r},t+\Delta t)$ is obtained, the algorithm is applied to the new profiles and the process is repeated. By continuing this procedure, the profiles are modified numerically through time in increments of Δt.

The computer program is written in Turbo Pascal Version 5.5 and a general flow diagram is shown in Figure 4.2. A system of subprograms is employed to calculate plasma resistance and inductance at each time step and return the values to the main program. An electrical subprogram is used in place of GCAP, which calculates the values of the electrical state variables (storage capacitor voltage and circuit currents) in the usual way. The value of the electric field across the plasma is returned to the main program and is used to update the electron temperature. Rate constants for the chemical processes are calculated and the new species densities determined. The updated plasma resistance and inductance are estimated. In this way, a detailed number and energy balance is maintained. This method is described more fully in Paper No.4 in Appendix C.

The time increment of calculation is automatically taken as one-fifth of the smallest time constant of the A-matrix. This is typically of the order of 50 psec. However, the rate equations represent an extremely stiff set of differential equations: the characteristic times of processes during the discharge pulse and in the immediate afterglow are very much shorter than those in the late afterglow. However, initial runs of the program confirm the observations of Bates et al[4] that the plasma afterglow experiences a quasi-steady state in which the electron density and temperature remain relatively constant. This allows the interpulse period to be divided into three time periods:

(1) the discharge current pulse;

(2) the immediate afterglow or quasi-stationary period;

and (3) the late afterglow.

The program reduces computer run-times by automatically changing the time

increment of calculation from the A-matrix if all parameters remain relatively constant for a time much greater than the time increment.

4.3 LONGITUDINAL ELECTRODE CONFIGURATION RESULTS

This thesis is concerned with the performance of a strontium vapour laser in response to parametric variation of the excitation circuit. This section considers a quasi-stationary uniform plasma consisting of helium and strontium particles and provides an insight into the development of the plasma. Due to the large number of inter-dependent variable parameters employed in the computer model, it becomes difficult to identify the effects of varying a single parameter. The dependence of laser output power and particle number densities on variable circuit parameters - charging voltage, discharge capacitance and pulse repetition frequency - are discussed. Also considered are the effects of varying the thermodynamic parameters of buffer gas pressure and strontium vapour pressure. The latter is a function of pulse repetition frequency and thermal design of the laser head. Equipped with the results of this analysis, and the basic understanding of SVL performance that this provides, the effect of adding neon to a quasi-stationary uniform helium-strontium plasma and the effect of varying the ratio of helium to neon in the buffer gas mixture is demonstrated.

For the purpose of comparing the model predictions with experimental data, parameters are chosen which accurately represent those encountered in the experiment described in Chapter Five. The default parameters used in the computer model are listed in Table 4.1. The construction of the laser is described in Chapter Five.

	Value	Units
Electrode spacing (l_d)	50	cm
Active zone length (l_a)	45	cm
Optical cavity length (l_c)	150	cm
Output mirror reflectivity ($\mathscr{R}$)	30	%
Tube radius (R_d)	13	mm
External helium pressure (p_{He})	100-750	torr
Percentage of neon in helium-neon mixture (%Ne)	10	%
Tube wall temperature (T_w)	550-750	^oC
Supply voltage (V_s)	0-12	kV
Storage capacitance (C_s)	8	nF
Smoothing capacitance (C_1)	1	μF
Charging inductance (L_c)	150	mH
Thyratron inductance (L_t)	200	nH
Bypass inductance (L_b)	120	μH
Bypass inductor resistance (R_b)	1	Ω
Pulse repetition frequency (prf)	0.1-3.0	kHz

Table 4.1. Standard parameters for longitudinal SVL model

4.3.1 STANDARD OPERATING PARAMETERS

Prior to switching of the thyratron, measurements based on Hook spectra reported by Künnemeyer et al[1] under experimental conditions of charging voltage, capacitance, and buffer gas pressure similar to those considered in this model, indicate that at a pulse repetition frequency of 3.77kHz, 3% of the strontium vapour remains ionised. In the model described here, the ground state populations of SrI and SrII are chosen to be $2.4\text{x}10^{14}\text{cm}^{-3}$ and $4.0\text{x}10^{12}\text{cm}^{-3}$ (1.6% ionisation). The pre-pulse electron density is also set at $4.0\text{x}10^{12}\text{cm}^{-3}$.

Computed voltage and current waveforms for a discharge in 200mbar of helium

at a wall temperature of 625°C are shown in Figure 4.3. The 4nF storage capacitance is charged to 14.4kV. As the storage capacitor voltage is transferred to the laser head, the plasma resistance drops from a value of 11.5kΩ to a minimum of 80Ω. The peak current is 90A, and the current pulse-width (FWHM) is 350ns. These values agree reasonably well with the experimental values described in Chapter Five. At peak laser current, the plasma resistance is 98Ω. A slight impedance mis-match exists between the excitation circuit and the laser load. Residual current ringing may be removed in practice by inserting a fast-recovery solid-state diode in parallel with the laser head[5]. The diode provides an alternative, low-impedance path for inverse current.

In Figure 4.4, the longitudinal electric field and the electron temperature are plotted as a function of time. Following switch closure (t=0), the electric field across the laser head rises as energy is transferred from the storage capacitor bank to the laser head. The electric field in the plasma reaches a maximum of 280Vcm^{-1} after approximately 60ns. During the rise-time of the discharge current pulse, the dominant source term for the electron temperature is that due to the electric field (equation (3.58)). The electron temperature follows the electric field with a time lag of the order of 20ns. The breakdown electron temperature is 3.3eV. Also shown in Figure 4.4 is the re-heating of the electron distribution which would result from an impedance mis-match between the excitation and load circuits in the absence of a fast recovery diode.

As the electric field diminishes, the electron temperature decays as shown in Figure 4.5. Initially, the decay is rapid, due primarily to electron thermalisation by elastic collisions with heavy particles. This coupling of the gas temperature to that of the electrons is responsible for gas heating

(Equation 3.61). The rate at which electrons cool slows down at about 700ns for two reasons. First, the rates of excitation and ionisation are very much reduced at temperatures below approximately 0.5eV; the elastic collision loss term of equation (3.58) is less effective. Secondly, electron-helium metastable superelastic collisions and helium/strontium Penning ionisations and charge transfer reactions reheat the distribution.

Figure 4.6 shows the development of helium species number densities with time. The decrease in the rate of decay of electron temperature at $t \simeq 700$ns coincides with the decay of metastable helium. This effect is also predicted in an analysis of the copper chloride laser[6]. The electron temperature is maintained at 3000-4000^{o}K, and does not decay towards the gas temperature until a few microseconds into the afterglow. As explained below, this behaviour is not detrimental to laser action. The electron density increases to a peak of $6.0\text{x}10^{14}\text{cm}^{-3}$ 600ns after switch closure and does not decay until the electron temperature becomes asymptotic to the gas temperature. Due to the relatively low ionisation potentials of the strontium species present, approximately 50% of the electron density in the afterglow of the current pulse is due to ionisation of SrI and SrII, and not helium (Figure 4.7); the fractional ionisation of helium never exceeds $2\text{x}10^{-4}$.

The population densities of strontium I, II, and III ground state species during the discharge current pulse and immediate afterglow are shown in Figure 4.8. The initial population of the SrI ground state is $2.4\text{x}10^{14}\text{cm}^{-3}$. The SrII ground state is populated from the previous discharge pulse to $4.0\text{x}10^{12}\text{cm}^{-3}$. As the electron temperature increases, the population of the SrII level increases due to electron-impact ionisation of SrI. The population of the SrI ground state decreases by two to three orders of magnitude over the period of a

discharge current pulse, implying that the strontium is >99% ionised. This is followed by electron-impact excitation to the Sr^{+*} levels (SrII excited states) and in turn by stepwise excitation to the Sr^{++} level (SrIII ground state), resulting in a decrease in the SrII ground state population during the period 400 - 700ns after thyratron switching. The qualitative nature of these observations is confirmed by the spectroscopic measurements of Künnemeyer et al[1]. A large population ($2.4x10^{13}cm^{-3}$) of Sr^{++} ions are accumulated during the discharge current pulse. This population is maintained for the remainder of the current pulse since the only loss mechanism - collisional-radiative recombination - is characterised by a rate coefficient which has a dependence of $Te^{-9/2}$ and is ineffective at the high electron temperatures encountered. Once the electron temperature drops below about 1eV, the plasma is characterised by a strong downward flux of species through the SrII excited states. The SrII pseudo-state, Sr^{+*}, is populated by collisional-radiative recombination of Sr^{++} ions. The high electron density ensures rapid depopulation of the Sr^{+*} level by electron-impact de-excitation, which populates the upper laser level, $6^2S_{1/2}$ (Figure 3.2). The lower laser levels $5^2P_{3/2,1/2}$ are similarly depopulated. Furthermore, since the electron temperature is sufficiently low to discourage further electron-impact excitation of the $5^2P_{1/2,3/2}$ levels from SrII states lying lower in energy, favourable conditions exist for the establishment of a population inversion in the early afterglow of the current pulse.

Figure 4.9 illustrates the temporal development of the $6^2S_{1/2}$ and $5^2P_{3/2}$ SrII level populations during the discharge current pulse and in the immediate afterglow. The behaviour of the electron temperature is shown in Figure 4.5 for reference. Notice that at no time does the density of ions in the $6^2S_{1/2}$ state exceed the density of ions in the $5^2P_{3/2}$ state. The inversion is partial, a result of the favourable difference in statistical weights of the two levels.

Shortly after the electric field has decreased to almost zero (t≈700ns), the populations of the two levels begin to increase for a second time due to the recombination pumping. At this time, T_e is approximately 3000K. At t≈800ns, a population inversion is achieved between the $6^2S_{1/2}$ and $5^2P_{3/2}$ levels. Figure 4.10 shows the development of laser intensity on the $6^2S_{1/2}$-$5^2P_{3/2}$ transition which has a risetime of about 50 ns. The laser pulse occurs immediately following termination of the current pulse at 800ns after switch closure and has a risetime of 100ns. The decay of laser intensity is characteristic of recombination. The stimulated emission pulse width (FWHM) is 300ns. Also shown in Figure 4.10 is the stimulated emission pulse in the presence of current ringing. The rapid termination of the laser pulse is due to the re-heating of the electron temperature (Figure 4.4).

Note that lasing on the $6^2S_{1/2}$ - $5^2P_{1/2}$ transition in SrII (λ=416.2nm) is suppressed for two reasons. Firstly, the ratio of the statistical weights of these two levels is unity. Secondly, the A coefficient for this transition is less than one half of that of the laser transition (λ=430.5nm).

4.3.2 INFLUENCE OF THE CAPACITOR CHARGING VOLTAGE ON THE GAIN CHARACTERISTICS OF THE ACTIVE MEDIUM

The computer model is run for values of capacitor charging voltage of 14.4, 16.0 and 20.0kV; all other parameters (C_s=4nF, PRF=1.0kHz, T_w=625^oC, p_{He}=200mbar) remain constant. The energy stored on the capacitor at the end of each charging cycle varies from 0.4 to 0.8J. The development of discharge current at each voltage is shown in Figure 4.11. As the initial voltage on the capacitor is increased, plasma breakdown occurs at an earlier time, thus leading to current pulses of greater amplitude and shorter duration.

A 100% increase in the peak density of Sr^{++} ions (SrIII) is observed from $2.0x10^{13}cm^{-3}$ at V_{Cs}=14.4kV to $4.0x10^{13}cm^{-3}$ at V_{Cs}=20.0kV. This is illustrated in Figure 4.12 and is due to the corresponding increase in energy stored in the capacitor. At a charging voltage of 20.0kV, the peak electron temperature is 45,000K, which is 15% above the value achieved at 14.4kV (Figure 4.13). At 20.0kV, the electron temperature remains near to its peak value for a shorter time than at 14.4kV; the rate of fall of electron temperature shows an increase from $0.75x10^{11}$ Ks^{-1} at 14.4kV to $1.3x10^{11}$ Ks^{-1} at 20.0kV. At 20.0kV, the rapid fall in electron temperature ensures a strong recombination flux and favourable conditions exist for the establishment of a population inversion. However, the increase in impedance mis-match between the excitation circuit and load encountered at the highest value of V_{Cs} produces an increase in the magnitude of the current overshoot (Figure 4.11). Figure 4.13 illustrates that, for a capacitor charging voltage of 20.0kV, the electron temperature drops to a minimum of approximately 5000K followed by a subsequent gradual rise in T_e due to current ringing. No population inversion is achieved at the higher values of V_{Cs}.

It is concluded that under the standard operating conditions listed in Table 4.1, a population inversion is established on the $6^2S_{1/2}$ - $5^2P_{1/2}$ transition in SrII (λ=416.2nm) on the achievement and maintenance of a low value of electron temperature ($T_e \simeq$3000-4000K) in the early afterglow of the discharge current pulse. An impedance mis-match between the modulator circuit and gas discharge load results in ringing of the discharge current pulse and subsequent re-heating of the electron distribution which in turn inhibits laser action.

4.3.3 INFLUENCE OF THE STORAGE CAPACITANCE ON THE CHARACTERISTICS OF THE DISCHARGE CURRENT

Section 4.3.2 demonstrated the effect on the gain characteristics of the SVL of varying the energy deposited in the gas by means of varying the capacitor charging voltage. For otherwise fixed conditions, variations in the energy deposited in the medium may also be achieved by varying the value of the storage capacitance, C_s. With the charging voltage held constant at 14.4kV, values of C_s of 4, 6 and 8nF are considered. The effect on discharge current is shown in Figure 4.14. Although increases in charging voltage result in a decrease in discharge current duration (Figure 4.11), an increase in current duration is observed at higher values of storage capacitance. This is explained by the fact that the rate of discharge of C_s in the absence of inductive effects is given by

$$\frac{dV_{Cs}}{dt} = -\frac{V_{Cs}}{RC_s} \qquad (4.6)$$

Hence, the larger the value of C_s, the longer a high voltage is sustained across the discharge tube. Figure 4.14 indicates that impedance matching is improved for C_s=8nF compared with C_s=4nF and, consequently, there is less re-heating of the electrons due to current ringing.

In summary, as the storage capacitance is increased, a large number density of Sr^{++} ions is created. The electron temperature falls towards 3000-4000K during the decay of the current pulse and there is little electron heating in the afterglow. A doubling of the energy stored in the discharge circuit achieved by doubling the value of storage capacitance from 4 to 8nF therefore appears to be a more favourable option for recombination than that achieved by an increase in charging voltage.

4.3.4 INFLUENCE OF THE CAVITY WALL TEMPERATURE ON THE GAIN CHARACTERISTICS OF THE ACTIVE MEDIUM

The results of Section 4.3.3 indicate that an increase in storage capacitance from 4 to 8nF improves impedance matching and leads to a greater rate of formation of Sr^{++} ions and to minimal re-heating of the electron distribution in the afterglow. However, in order to maintain the temperature of the active medium within a narrow range, the increase in deposited energy ($= \frac{1}{2} C_s V_{Cs}^2$) which results from the use of a higher storage capacitance must be accompanied by an increase in the rate of removal of heat from the laser head. The model described in this thesis does not include the loss of energy due to thermal conduction to the tube walls and consequently only sources of energy contributing to an increase in gas temperature are included. For a fixed laser head design, an increase in the wall temperature is produced by increasing the value of one or more of the storage capacitance, charging voltage or pulse repetition frequency (PRF). An increase in the PRF produces an an increase in the average power deposited into the active medium, but does not affect the energy deposited per pulse.

This section considers the increase in wall temperature resulting from an increase in the PRF. The population of the ground state SrI at time t=0 is calculated on the basis of a strontium vapour pressure relevant to the wall temperature[3]. The fractional ionisation of strontium at time t=0 is based on the spectroscopic measurements of Künnemeyer et al[1] for experimental conditions of PRF, storage capacitance, charging voltage, buffer gas pressure and laser head dimensions similar to those included in this model. Although the SrI ground state density increases with wall temperature, the fractional ionisation

of SrI at time t=0 is maintained constant at 1.6%.

The effect of increasing cavity temperature on discharge current, shown in Figure 4.15, is similar to that associated with an increase in charging voltage, ie the current pulse shows an increase in magnitude and a decrease in duration; an increase in cavity temperature from 625°C to 725°C causes an increase in current magnitude from 90A to 330A, whilst the pulse duration (FWHM) decreases from 350ns to 100ns. Current ringing is removed by means of a fast-recovery diode. The effect on electron temperature is illustrated in Figure 4.16. Also shown in Figure 4.16 is the development of stimulated emission intensity at the various values of cavity temperature. The peak value of the small-signal gain coefficient, g_o, is a maximum of 4.3%cm^{-1} (based on a Doppler broadened linewidth of the laser transition of 150MHz) at 675°C. The magnitude of g_o decreases at 625°C and does not exceed zero at 725°C. Consequently, lasing is not observed at this elevated temperature. It is concluded that the optimum cavity temperature for lasing lies in the range 625°C to 675°C.

4.3.5 INFLUENCE OF THE PRESSURE OF THE BUFFER GAS ON THE CHARACTERISTICS OF THE DISCHARGE CURRENT PULSE

Figure 4.17 illustrates the effect of increasing the buffer gas (helium) pressure from 200 to 300 mbar on the characteristics of the discharge current pulse. The remaining parameters are standard and are listed in Table 4.1. Although increased buffer gas pressure results in improved impedance matching, the peak value of discharge current is only 40A and the peak value of the Sr^{++} population density achieved is a factor of two lower than that at 200 mbar. Coupled with the decreased rate of decay of the current pulse, it is concluded that for the standard operating parameters listed in Table 4.1, an increase in

buffer gas pressure from 200 to 300 mbar results in a lower magnitude, and slower subsequent decay, of the electron temperature, which tends to inhibit population inversion.

4.3.6 ADDITION OF NEON TO THE BUFFER GAS MIXTURE

Several authors[7,8] have reported that addition of neon to the helium buffer gas/strontium vapour mixture improves the stability of the discharge. Of greater importance, regarding the recombination characteristics of the active medium, is the fact that, under certain experimental conditions, replacement of 10-20% of the helium buffer gas by neon increases laser output power. Three effects are observed[7] with increasing fractional neon concentration beyond this figure:

(1) The average laser output power decreases,

(2) the laser pulse width decreases

and (3) the time delay between termination of the discharge current pulse and the peak of the stimulated emission pulse increases.

A feature of the kinetics model described here is the facility to vary the ratio of helium to neon concentration in the buffer gas mixture. The distribution of excited states in a helium-neon mixture is well-documented with reference to both CW and pulsed He-Ne laser systems (Refs. 34 - 46, Appendix B). Simulations are performed in this section in which the percentage of neon in a helium-neon buffer gas mixture at a total pressure of 200 mbar is varied from 0% (pure helium) to 100% (pure neon). Figure 4.18 shows the computed current waveforms obtained in the presence of a fast-recovery diode for the case of a storage capacitance of 4nF charged initially to 14.4kV. As the fractional

concentration of neon is increased from 0% to 40% (0 to 80 mbar partial pressure), the peak magnitude of the discharge current pulse increases from 183A to 295A, whilst the duration (FWHM) decreases from 225ns to 180ns. The effect of increasing neon concentration is to lower the discharge impedance (Figure 4.19) due to the greater rate of ionisation of neon compared with that of helium (Figure 4.1a).

The reduced effectiveness of the elastic collision loss term in equation (3.58) for neon compared with helium results in a slower rate of decay of the electron temperature in the afterglow (Figure 4.20). Also shown in Figure 4.20 is the effect of increasing neon concentration on the temporal variation of stimulated emission. Maximum laser output power is obtained with pure helium buffer gas. Addition of neon to the buffer gas results in a lower peak magnitude and more rapid decay of laser intensity than for pure helium at 200 mbar. The time delay from termination of the discharge current pulse to maximum stimulated emission at 430.5nm increases from 340ns in pure helium to 540ns in a 40%Ne-60%He buffer gas. These results agree with the experimental observations (1) - (3) above. The laser pulse width decreases for increasing neon concentration since the rate of fall of electron temperature is decreased. The establishment of a population inversion is delayed in time, at which point the populations of both the upper and lower laser levels have fallen considerably.

Figure 4.21 illustrates the dependence of both peak and average laser output power on the fractional concentration of neon in the buffer gas mixture. Replacement of 20% helium with neon causes the average laser power to be reduced by 50%. The pulse duration (FWHM) drops from 170ns in pure helium to 145ns. For a 60%Ne-40%He buffer gas, no laser action is observed.

The simulations described above are confirmed by two sets of experimental results. Little and Piper[8] report results obtained for a 25mm id discharge tube. At a pressure of 200 mbar, average laser power is a maximum for a pure helium buffer gas and decreases to ≈30% at a neon concentration of 15%. McLucas and McIntosh[7] report laser characteristics for a 13mm id SVL obtained over a full range of helium-neon buffer gas mixtures at a total pressure of 190 mbar. Average laser output power and pulse width decrease with increasing neon concentration above 20%, and exceed zero up to a mixture of 80%Ne-20%He. The time delay between termination of the discharge current pulse and the peak of the stimulated emission pulse increases with increasing neon concentration from 320ns for pure helium. It is concluded that at a pressure of 200 mbar, addition of neon to the buffer gas does not increase either average or peak laser output power.

4.4 CONCLUSIONS

The results of the computer model SRRAD applied to analyse the effect on laser performance of varying electrical circuit parameters indicate the fundamental importance of impedance matching. For a laser head of fixed thermal design, there exists a set of optimum operating parameters. For the laser system considered here - a 13mm id, 50cm length discharge tube containing helium at a pressure of 200mbar - the optimum parameters are found to be:

Storage capacitance	:	4 nF
Capacitor charging voltage	:	14.4 kV
Cavity wall temperature	:	625-675^{o}C

A factor of two increase in the density of Sr^{++} ions is achieved by doubling

the energy deposited per pulse by means of increasing either the charging voltage or the storage capacitance. However, in the former case, impedance mis-match results in re-heating of electrons in the afterglow, inhibiting laser action. Improved matching is achieved by increasing the value of the storage capacitance. It is imperative, however, under these operating conditions that the cavity temperature is maintained within the range 625^{o}C to 675^{o}C. Ideally, as much electrical energy as possible should be discharged into the laser volume in order to create a large population of Sr^{++} ions, and the heat removed from the discharge as rapidly as possible in the afterglow.

In the following chapter, experimental conditions similar to those described in the simulations above are implemented.

REFERENCES FOR CHAPTER 4

1. R. Künnemeyer, C. W. McLucas, D. J. W. Brown and A. I. McIntosh, "Time-resolved measurements of population densities in a Sr^+ recombination laser", IEEE J. Quantum Electron., QE-23 (11), pp.2028-2032 (1987).

2. E. Jahnke and F. Emde, 'Tables of Functions with Formulae and Curves', Second edition, B. G. Teubner, p.194 (1933).

3. An. N. Nesmeyanov, 'Vapour Pressure of the Elements', MacMillan and Co. Ltd., pp.185-187 (1963).

4. D. R. Bates, A. E. Kingston and R. W. P. McWhirter, "Recombination between electrons and atomic ions. I. Optically thin plasmas", Proc. Roy. Soc., A267, p.297 (1962).

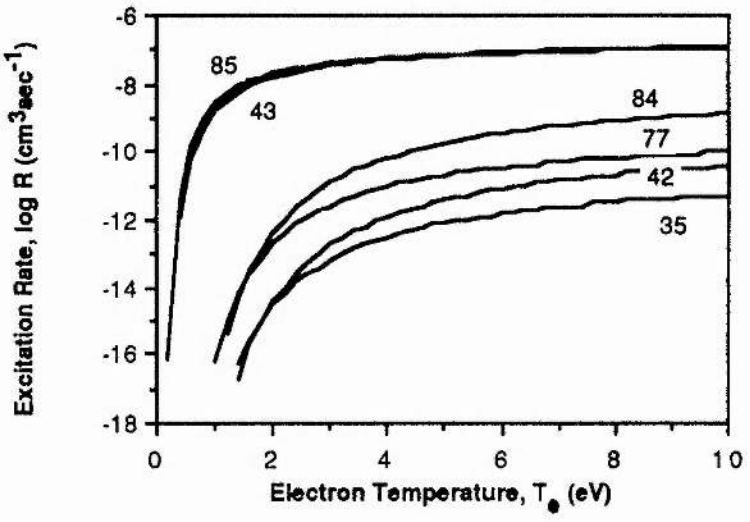
5. C. E. Little and J. A. Piper, "High-power violet Sr^+ recombination lasers", SPIE vol.1041, *'Metal Vapor, Deep Blue, and Ultraviolet Lasers'*, Los Angeles, CA, pp.167-174 (1989).

6. M. J. Kushner and F. E. C. Culick, "A model for the dissociation pulse, afterglow, and laser pulse in the Cu/CuCl double pulse laser", J. Appl. Phys., 51 (6), pp.3020-3032 (1980).

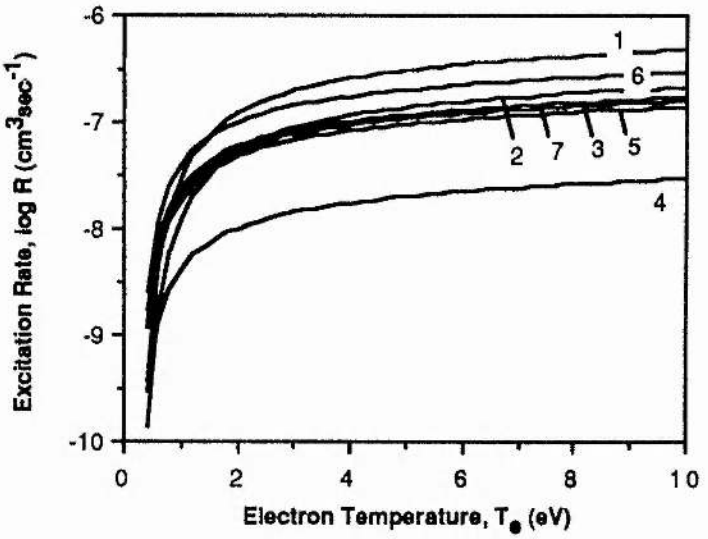
7. C. W. McLucas and A. I. McIntosh, "Investigation of laser emission in Sr^+ and Ca^+", J. Phys. D: Appl. Phys., 20, pp.591-596 (1987).

8. C. E. Little and J. A. Piper, "Average-power scaling of self-heated Sr^+ afterglow-recombination lasers", IEEE J. Quantum Electron., QE-26 (5), pp.903-910 (1990).

FIGURES FOR CHAPTER 4

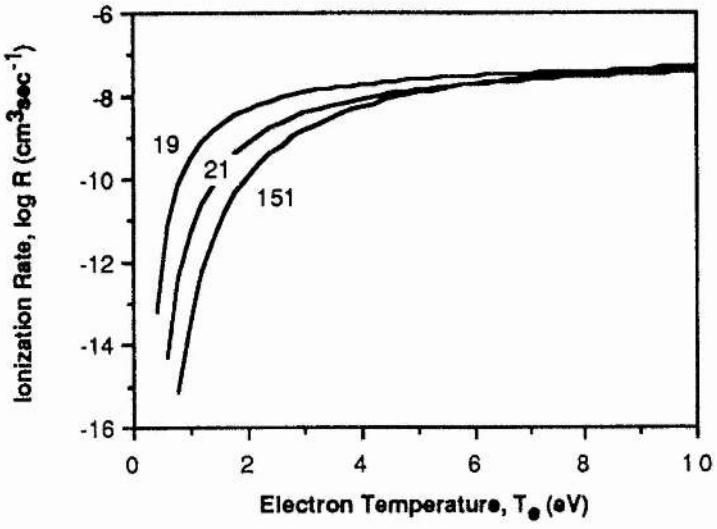


(a)



(b)

Figure 4.1. Electron impact excitation rates used in SRRAD
(a) Helium and neon
(b) Strontium.
Figures refer to rate processes listed in Appendix B



(c)

Figure 4.1 (Continued). Electron impact ionization rates used in SRRAD
(c) Strontium.
Figures refer to rate processes listed in Appendix B

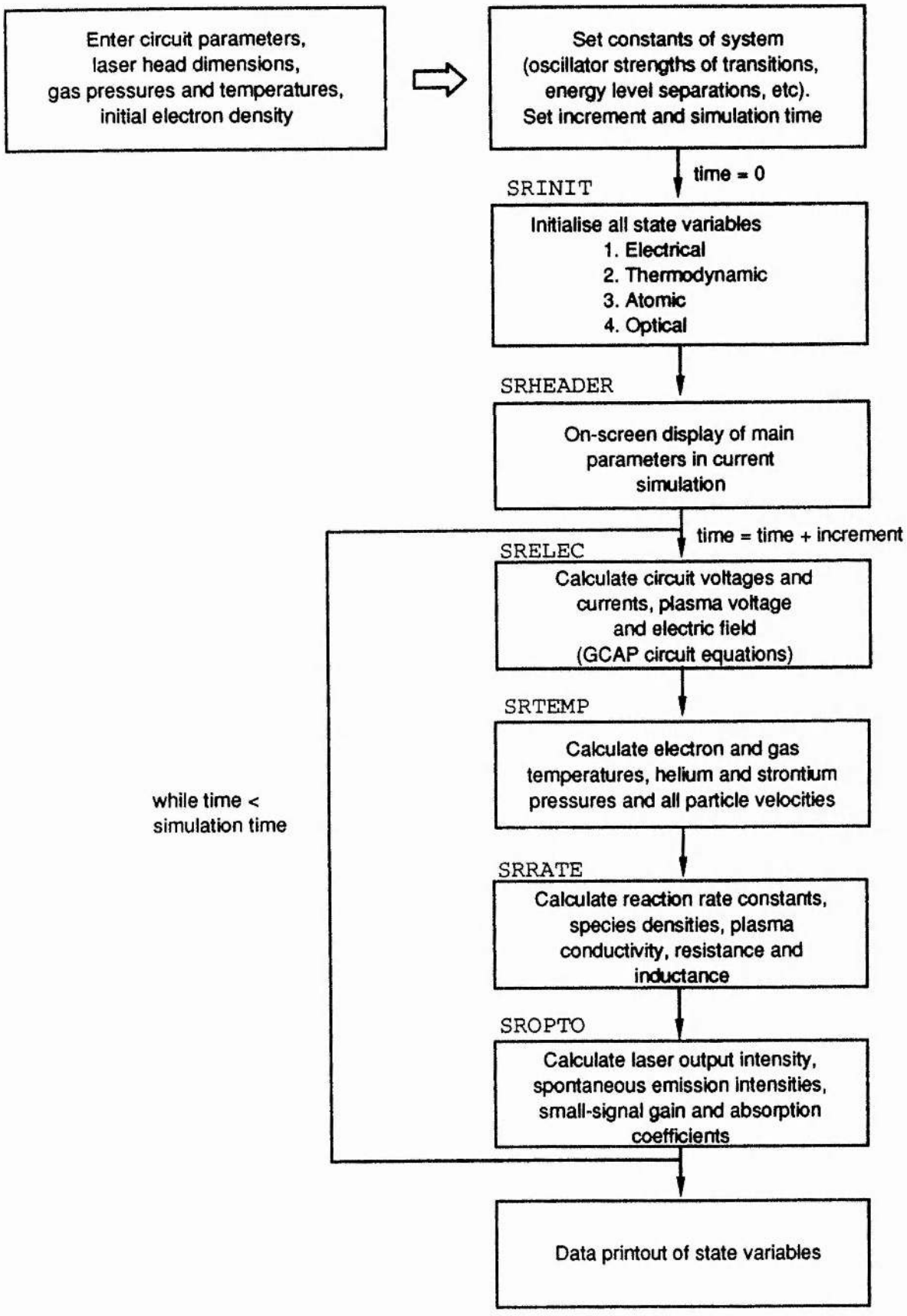


Figure 4.2. SVL computer model flow diagram

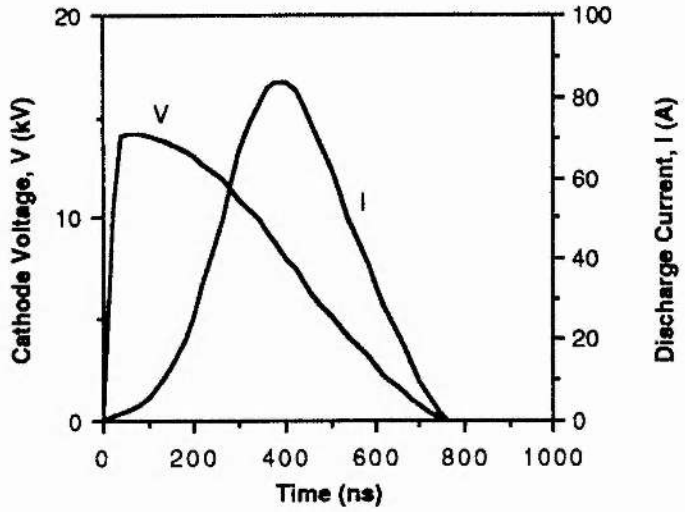


Figure 4.3. Computed voltage and current waveforms for a discharge in 200 mbar of helium with the parameters in Table 4.1 (V_o=14.4kV). Peak current is 90A and FWHM is 350ns. Typical experimental values are 60-180A and 200-400ns (Chapter Five)

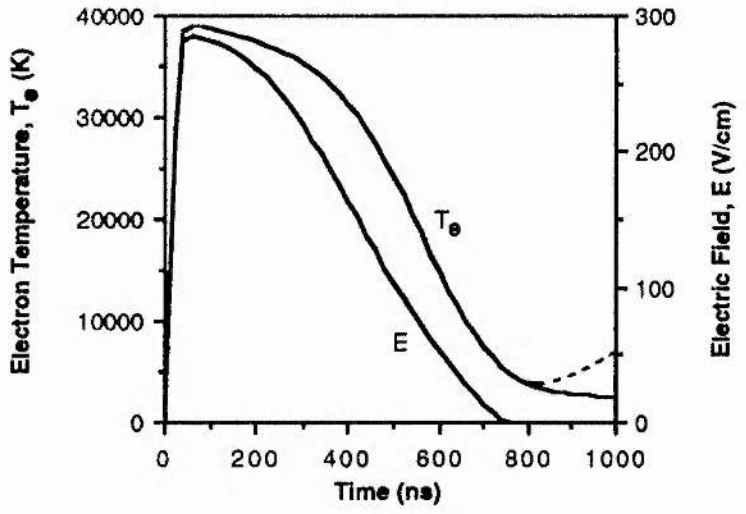


Figure 4.4. Electric field strength and electron temperature for a discharge in 200 mbar of helium in the presence (solid line) and absence (dashed line) of a fast-recovery diode. The superelastic electron heating is noticeable at t≈800ns

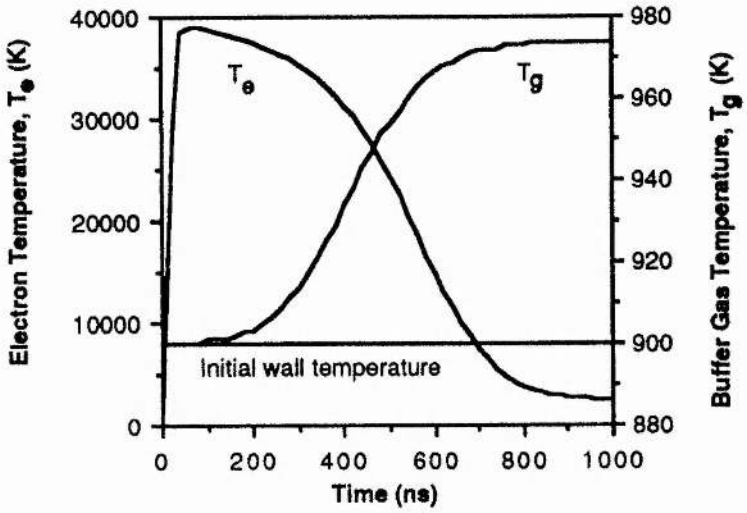


Figure 4.5. Electron and heavy particle temperatures. Note the different axis scales

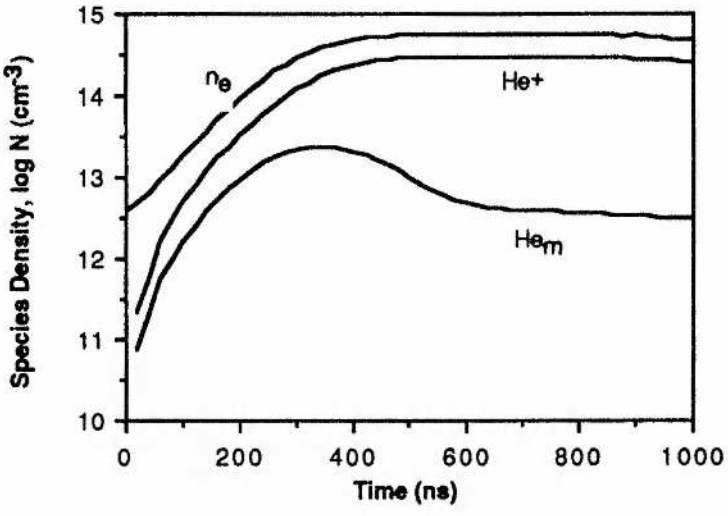


Figure 4.6. Density of electrons, helium ions and helium metastables for a discharge and afterglow in 200 mbar of helium and of helium and strontium. Note that the helium metastable is short-lived compared with the ion - it is rapidly depleted by Penning ionisations and charge exchange

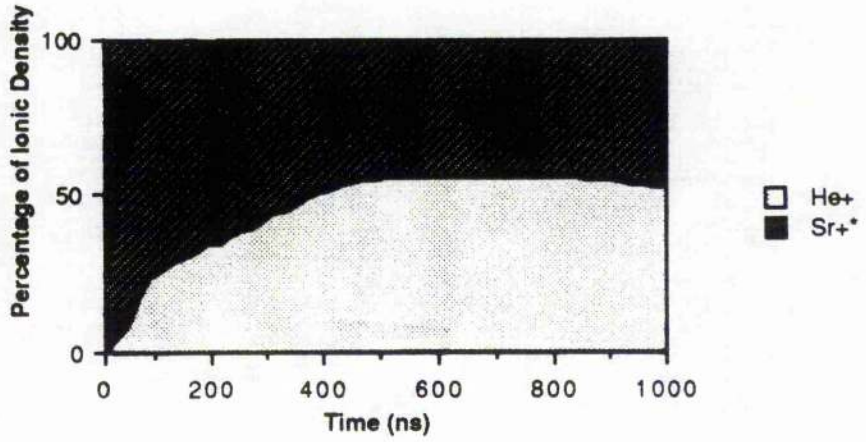


Figure 4.7. Percentage contributions to plasma conductivity as computed with the discharge pulse and afterglow model

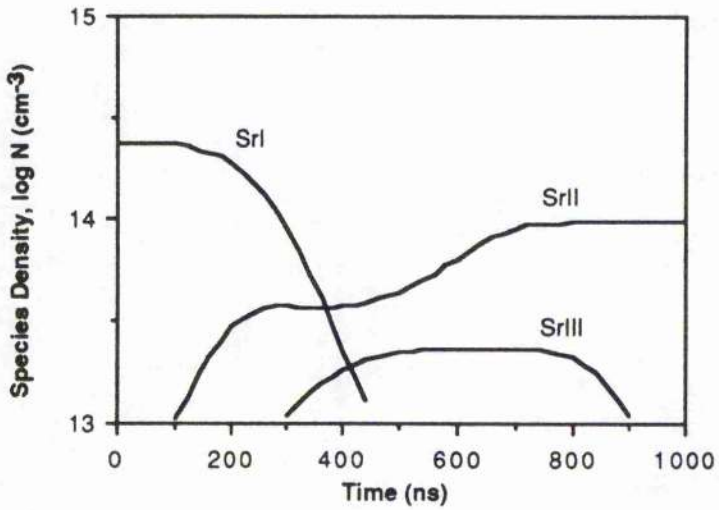


Figure 4.8. Density of SrI, SrII and SrIII species of strontium at its vapour pressure for T_g=625^{o}C

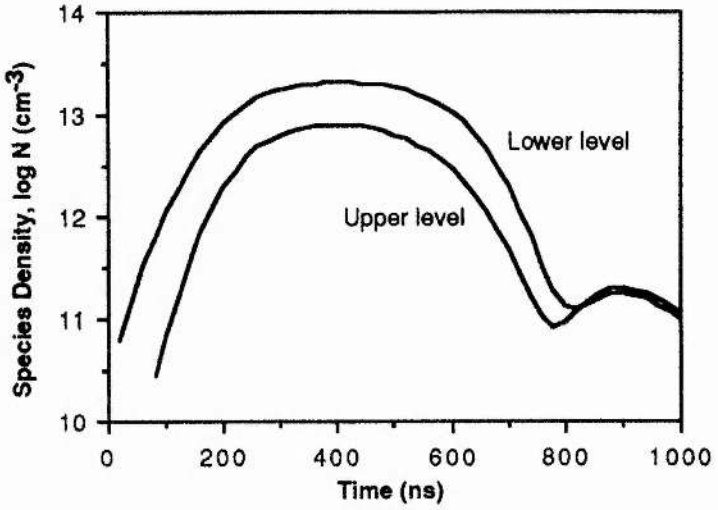


Figure 4.9. Discharge and afterglow in 200 mbar of helium and strontium at its vapour pressure for 625^{o}C. Note that the density of $6^2S_{1/2}$ exceeds half that of $6^2P_{3/2}$ after 800ns, at which time $T_e \simeq 3000K$

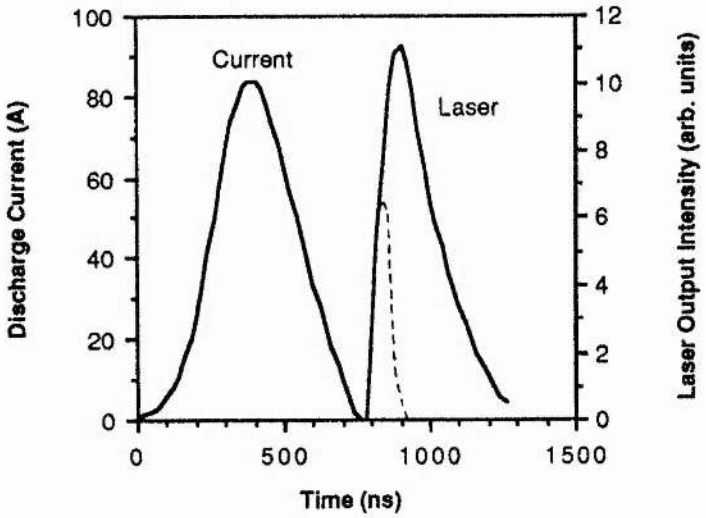


Figure 4.10. Discharge current and stimulated emission intensity as a function of time in the presence (solid line) and absence (dashed line) of a fast-recovery diode

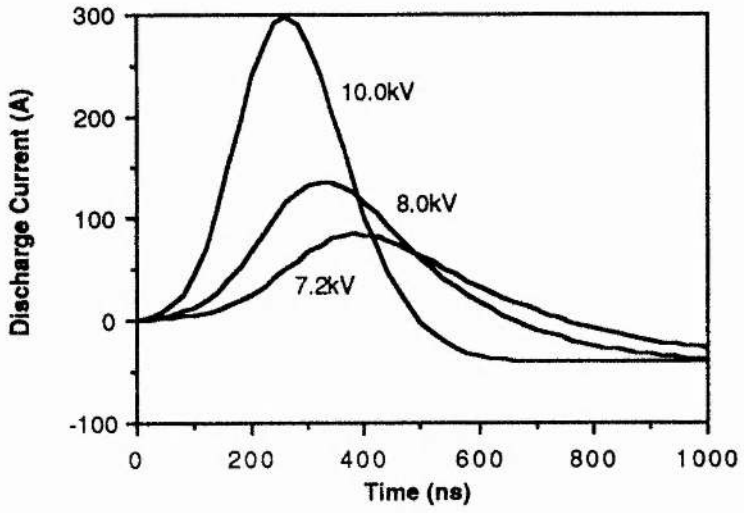


Figure 4.11. Effect of capacitor charging voltage on discharge current pulse: (a) V_{cs} = 7.2kV, (b) V_{cs} = 8.0kV and (c) V_{cs} = 10.0kV

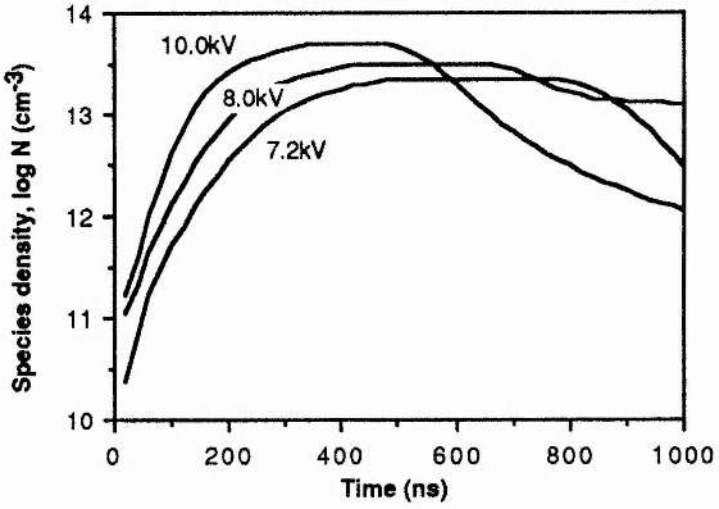


Figure 4.12. Density of the positive Sr^{++} ion as a function of capacitor charging voltage:
(a) V_{cs} = 7.2kV, (b) V_{cs} = 8.0kV and (c) V_{cs} = 10.0kV

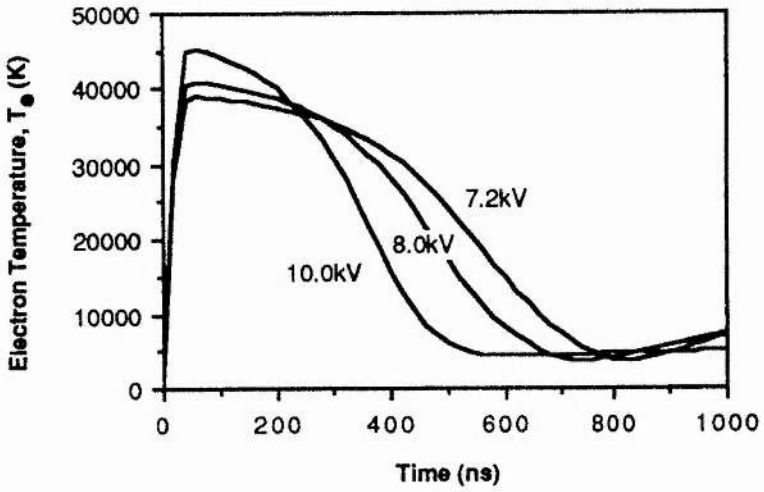


Figure 4.13. Development of electron temperature as a function of capacitor charging voltage:
(a) V_{cs} = 7.2kV, (b) V_{cs} = 8.0kV and (c) V_{cs} = 10.0kV

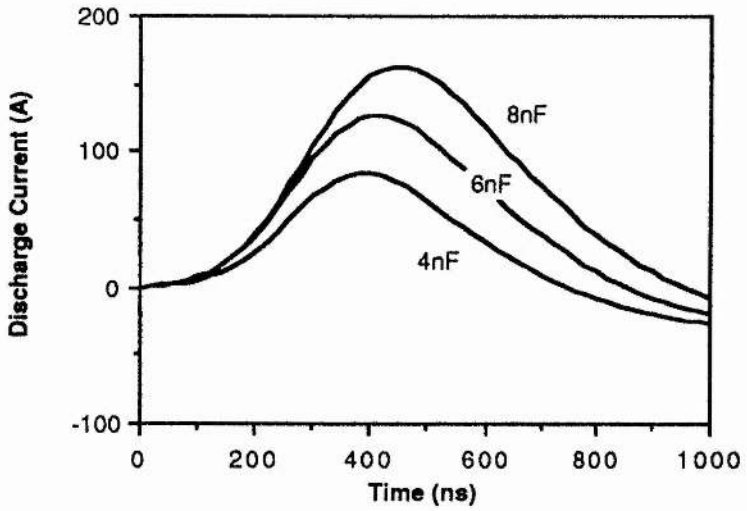


Figure 4.14. Effect of storage capacitor value on discharge current pulse:
(a) C_s = 4.0nF, (b) C_s = 6.0nF and (c) C_s = 8.0nF

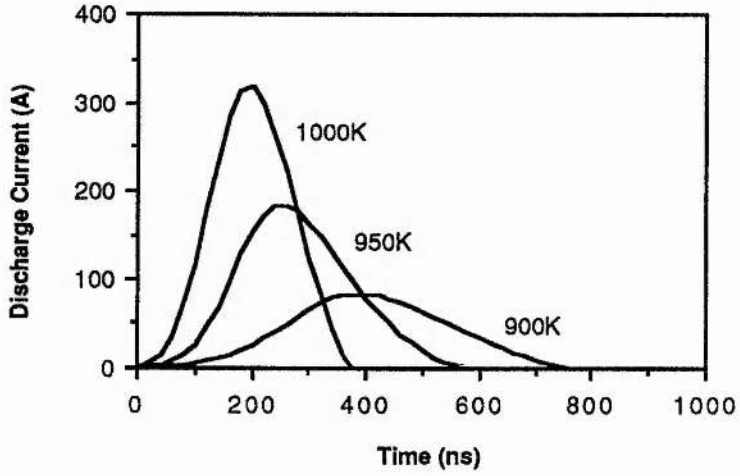


Figure 4.15. Effect of cavity temperature value on discharge current pulse in the presence of a fast-recovery diode:
(a) T_g = 900K, (b) T_g = 950K and (c) T_g = 1000K

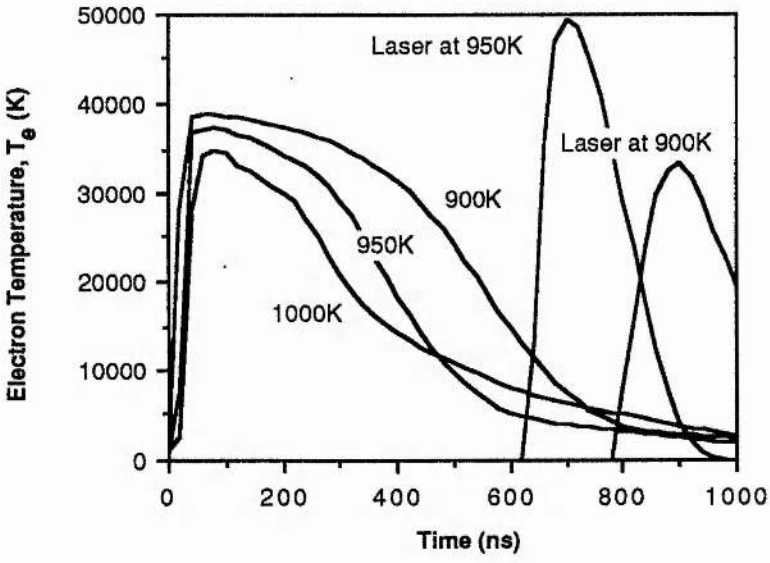


Figure 4.16. Effect of cavity temperature value on electron temperature and stimulated emission intensity:
(a) T_g = 900K, (b) T_g = 950K and (c) T_g = 1000K

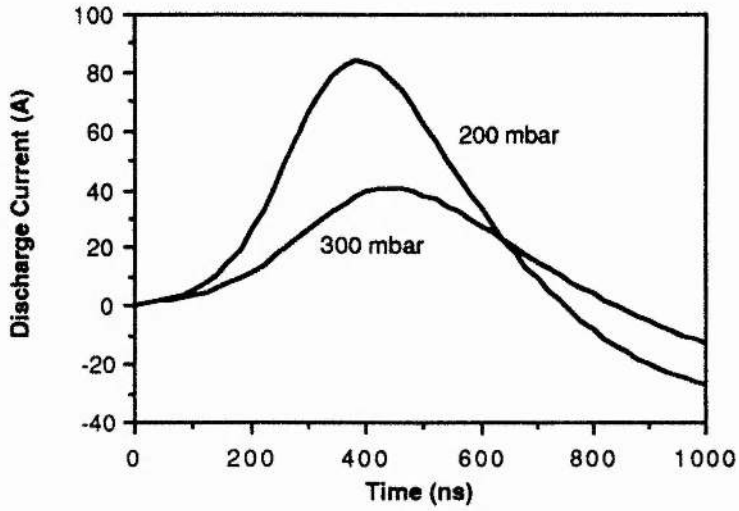


Figure 4.17. Effect of helium pressure value on discharge current pulse: (a) p_{He} = 200 mbar and (b) p_{He} = 300 mbar

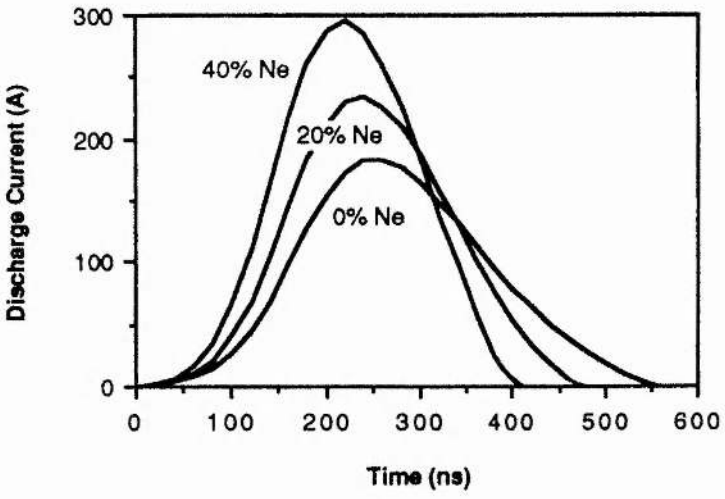


Figure 4.18. Computed current waveforms for a discharge in 200 mbar total pressure in a helium-neon-strontium vapour mixture. Current ringing is suppressed by the use of a fast-recovery diode. V_o=14.4kV. (a) 0% Ne, (b) 20% Ne and (c) 40% Ne

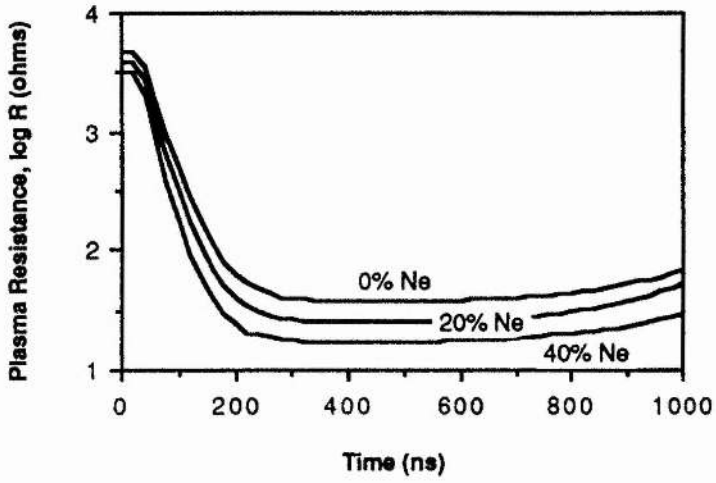


Figure 4.19. Development of plasma resistance as a function of percentage of neon in the discharge:
(a) 0% Ne, (b) 20% Ne and (c) 40% Ne

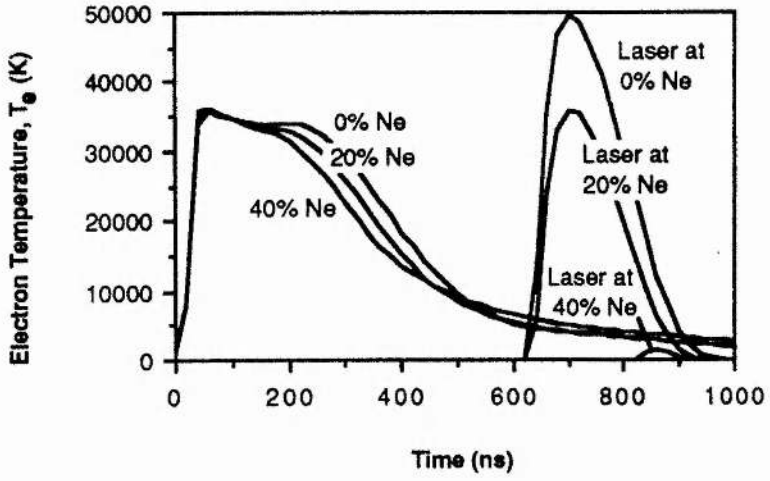


Figure 4.20. Development of electron temperature and stimulated emission intensity as a function of percentage of neon in the discharge:
(a) 0% Ne, (b) 20% Ne and (c) 40% Ne

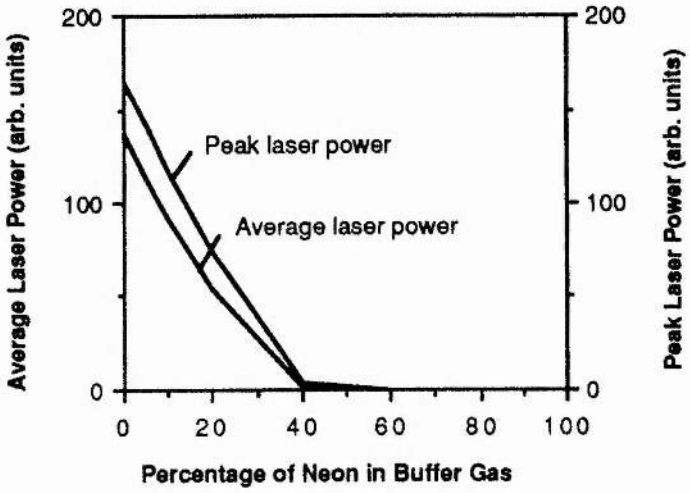


Figure 4.21. Dependence of average and peak laser power on percentage of neon in the helium-neon buffer gas

Chapter 5

Strontium Vapour Laser Experiments

5.1 INTRODUCTION

This Chapter describes the experimental apparatus, technique and results obtained with the strontium vapour laser (SVL). The laser system consists of a laser head and electrical circuit together with water cooling and gas handling systems. Emphasis is placed on the design of the pulsed power section of the SVL. The values of modulator circuit parameters such as storage capacitance and pulse repetition frequency are chosen to correspond to the optimum values for laser emission based on the state variable analysis of Chapter Four. Experiments conducted to study the effect on laser output power of variations in pulse repetition frequency, amplitude of the discharge current pulse and pressure of the buffer gas are described and the results are compared with the computer simulations.

5.2 EXPERIMENTAL APPARATUS

5.2.1 ELECTRICAL DISCHARGE CIRCUIT

The electrical circuit used to create a discharge is a thyratron-switched series capacitance circuit, shown in Figure 5.1. The design of each circuit component is described below.

5.2.1.1 Laser Power Supply

Description of Circuit Operation

The source of HT used in the laser experiments is a conventional, transformer-isolated power supply capable of providing 0.5A output at 21kV. The circuit arrangement is shown in Figure 5.2 and operation is as follows. Power enters the system by way of a circuit breaker, illuminating three 1W amber neon indicators. Closing the "HT OFF" switch energises relay D, which in turn causes contact $R_L D$ to open, thus preventing current flow. Providing there are no DC or AC overload conditions present on the line, then the green "ON" indicator is now lit. Closing the "HT ON" switch energises contactor B and closes contacts B1-3. Current now flows through the high-wattage resistors R1-3. Contact B5 now opens, de-energising relay D and closing contact $R_L D$. The time constant associated with the RC combination connected across this relay introduces a half-second delay in the closing of the contact. Contactor C is now energised, closing contacts C1-3 and shorting out resistors R1-3. This "soft-start" approach, whereby current is initially limited by resistors R1-3, avoids the problem of inrush currents magnetising the transformer windings at turn-on. The red "HT ON" neon indicator is now illuminated.

Voltage control is provided by the three-phase variac and the current is limited to 10 amps per phase by means of fuses. The delta-star three-phase transformer steps up the primary voltage to a maximum of $15kV_{rms}$, which is rectified by a full-wave diode bridge. The bridge consists of six arms, each containing two J-Ea 9000-0.7A rectifier diodes[1] connected in series. The total voltage hold-off capability of each pair of diodes is 18kV. The pulsating DC output voltage is smoothed by a 40kV, 1μF capacitor. The HV output is up to

21kV, with low voltage ripple, at 0.5A. A 100kΩ resistance consisting of an arrangement of ten series by two parallel ceramic 17W resistors is connected across the smoothing capacitor by means of a high-voltage switch. This enables the capacitor to be discharged when the power supply is turned off.

AC and DC Overload Protection

(a) AC Overload

The current flowing in the output arms of the variac passes through the coils a-b, c-d and e-f (Figure 5.2) which each constitute the primary windings of a 24V:0.24V, 20A transformer[2]. The secondary voltages are rectified and connected in series. The 10μF capacitor smoothes the DC output voltage. The variable resistance is adjusted manually such that relay H is energised when the DC voltage exceeds a pre-determined level. When this occurs, contact R_LH opens and the power supply output is disabled.

(b) DC Overload

The DC overload protection circuit is located on the earth side of the HT output of the power supply. The associated voltage-sense resistor is connected in series with the thyratron cathode. Common current develops a voltage drop across the resistor which, if it exceeds a pre-determined level, causes contact R_LJ to trip out, thus disabling the power supply output.

Monitoring

(a) HT DC Current

An ammeter of fsd 1A monitors the HT output current.

(b) Variac AC Voltage

The voltage applied to the primary windings of the three-phase transformer is monitored directly by means of a 0-300V_{AC} voltmeter.

(c) HT DC Voltage

The HT output voltage (up to 20kV_{DC}) is monitored indirectly on one of the low-voltage primary windings of the three-phase transformer. A diode bridge rectifies the AC voltage and a 2.7MΩ resistor in series with a 1300Ω coil resistance, 100μA fsd voltmeter converts the fsd to 300V_{AC}. This meter is calibrated to indicate the DC voltage present at the power supply output.

5.2.1.2 Resonant Charging and Laser Bypass Inductor

Design of a Resonant Charging Inductor

The element used to charge the PFN in Figure 5.1 may be either a resistor or an inductor. Since energy is stored and not dissipated in an inductor, inductive charging is the more efficient process. Furthermore, the voltage v(t) on the anode of the thyratron at a time t immediately following the discharge pulse is given by[3]

$$v(t) = V + \exp\left[-\frac{R}{2L}t\right]\left[v(0) - V\right]\left[\cos(\omega t) + \frac{R}{2\omega L}\sin(\omega t)\right] \qquad (5.1)$$

where V is the power supply voltage, R and L are the total charging circuit resistance and inductance, v(0) is the initial voltage on the PFN capacitance C and ω, the natural frequency of the charging circuit, is given by

$$\omega^2 = \frac{1}{LC} - \frac{1}{4L^2} \quad . \tag{5.2}$$

The voltage appearing on the thyratron anode increases slowly at the beginning of a cycle. The plasma within the tube has time to recover before the application of anode voltage.

If the charging inductance is chosen to have the value

$$L_R = \frac{1}{\pi^2 f^2 C} \tag{5.3}$$

where f is the switching frequency, then the voltage on the thyratron anode at the end of a discharge cycle of period T is found from equation (5.1) to be

$$v(T) = 2V - v(0) \quad . \tag{5.4}$$

Hence, if the voltage on the capacitor C is zero at the beginning of a charging cycle, then it is possible to charge the PFN up to twice the power supply voltage. This method is known as resonant charging. If the value of inductance chosen is less than that given by equation (5.3) or, equivalently, if the circuit is run at a lower operating frequency, then the inductor current goes negative and the charging voltage decreases. Current reversal is prevented by inserting a diode in series with the charging inductor. In addition, if the charging inductance is made too small, the peak charging current in the inductor, $\hat{i}$, which is given by

$$\hat{i} = V \left[\frac{C}{L}\right]^{0.5} \tag{5.5}$$

increases, resulting in the possibility of core saturation. Saturation is

avoided by inserting an air gap into the magnetic flux path.

If the charging inductance value exceeds that given by equation (5.3) then a more linear increase in voltage is produced which may inhibit thyratron recovery.

The resonant charging inductor is constructed according to the principles described above. The core consists of four ferrite U-cores, each of dimensions 102 x 25 x 57 mm (length x breadth x height). For a resonant frequency of the charging circuit of 8kHz, and for a discharge capacitance of 8nF, the value of the charging inductor is 200mH. The peak current is calculated from equation (5.5) to be a maximum of 4A (for a 20kV charging voltage). In order to prevent saturation of the magnetic cores, a total of four layers of Mylar are inserted in the magnetic path (Figure 5.3). Five layers of forty turns per layer of 1.25A insulated wire are wound on a reinforced nylon coil former. Since the layer ends will "see" a potential difference of up to 16kV, two thicknesses of Mylar (10kV voltage hold-off per thickness) are inserted between each layer. Heat-shrink insulation is applied to the low and high voltage connections. The two ferrite cores are held together by means of a clamp consisting of two flat lengths of wood connected by two threaded rods (Figure 5.3). Tightening of the rods provides a limited range of inductance adjustment once the design is completed.

The charging diode is a silicon diode (No. KHP40[4]) rated to 2.25A average forward current at a peak reverse voltage of 40kV. The charging diode must be rated to withstand a peak reverse voltage of at least twice the full network voltage since at the end of its charging period, the thyratron anode may swing down to earth potential unless the circuit is adequately damped. Extra current

damping is provided by connecting an RC snubber circuit across the charging inductor as shown in Figure 5.1. Transient voltage spikes reflected from the discharge circuit are prevented from reaching the charging diode by means of a parallel inductor-resistor combination connected between the discharge circuit and the diode.

Laser Bypass Inductor

The value of the laser bypass inductance is chosen so as not to have an adverse effect on the capacitor charging time. However, if the value is too low, the inductor will draw a significant part of the current during the discharge period. If the impedence of the bypass element is high enough, an insignificant amount of the discharge current will flow through it. A range of values of inductance were tested, including a firebar. A value of 120 μH was chosen as this gave the best trade-off between magnitude of laser current and pulse duration. This inductor consists of 120 turns of 5A insulated wire wound in three layers separated by heatshrink insulation on a 40mm o.d. nylon former.

5.2.1.3 Switching Element

The characteristics of a switching element of greatest interest in a capacitive-discharge circuit providing short pulses into a varying impedance load are those of peak current rating and rate of rise of current. The switching element used in the capacitor-discharge circuit of Figure 5.1 must be able to withstand voltages of 10-30kV and pass peak and average currents of 1kA and 0.25A, respectively, with a rate of rise of current of 5 kA/μs at a pulse repetition frequency of between 1kHz and 30kHz. Consideration of Table 5.1 indicates that this immediately removes the possibility of employing single

solid-state switches in this role. For power MOSFETs, the peak current is limited to approximately 200A for a single device[5]. However, this limitation may be overcome by series-parallel arrangements. An example of this is a 0.5MW, 60kHz power modulator built at Maxwell Laboratories[6] which passes 700A peak current at 5.5kV. The rate of rise of current is 1.4 kA/μs. Silicon-controlled rectifiers (SCRs), on the other hand, can handle pulsed currents up to 34kA[7] (of duration 8.3ms) but suffer from limitations on dv/dt and di/dt. The latter is typically 100 A/μs which practically eliminates consideration of their use as a switch in the laser modulator. More recently, the reverse-conducting thyristor[8], whose gate structure has been optimised for fast turn-on, can handle rates of rise of current in excess of 20 kA/μs. A series connection of four of these devices has produced current pulses in excess of 2kA of duration 400ns with a switching frequency of 5kHz and shows promise as a high frequency, high power solid-state switch.

Device	Type	$V_{hold\text{-}off}$ (kV)	I_{av} (A)	I_{pk} (kA)	di/dt ($x10^{10} As^{-1}$)	Reference
Power MOSFET	2N7070	0.5	50	0.2	0.2	5
SCR	CS1104 -55lo1	5.5	2000	34	0.1	9
Spark gap	High pressure	1000	Low	1	1000	10
Thyratron	CX2025	60 (@4.5kHz)	3.4	>3	5	11
	CX1826	35	5.0	5	5	9

Table 5.1. A comparison of high power switching parameters

A thyratron is used as the main switch in the capacitor-discharge circuit of Figure 5.1. In such an application, in which the inductance is minimised, the

thyratron is required to deliver short pulses into a load whose impedance can vary by several orders of magnitude. Peak forward currents and rates of rise of current required are in excess of 10kA and 100 kA/μs respectively[12]. The reliability, voltage hold-off capability, peak current rating and ability to operate at frequencies in the multi-kilohertz regime have made the hydrogen thyratron the best candidate for use in modulator circuits for short pulses, in particular for metal vapour and repetition-rated excimer lasers. Two thyratrons - the EEV CX1535 and CX1826 - are investigated in this project. An explanation of the operating principles of the thyratron is given in the following section, followed by a description of the CX1826 and its associated triggering circuitry.

The Hydrogen Thyratron

The hydrogen thyratron is a high-voltage, high peak current, repetitive closing switch. The thyratron essentially consists of three main components: the anode and a thermionic cathode separated by a baffled grid such that there is no direct path between the two electrodes. These electrodes are housed within a ceramic or glass envelope which is filled with a light gas (hydrogen or deuterium) at low pressure. The thyratron is capable of switching tens of kiloamps when triggered by a voltage pulse of a few hundred volts. The operation of the thyratron is discussed in reference 3.

Two thyratron types are used in the SVL experiments. The first of these is a CX1535A ceramic type. However, the maximum anode voltage hold-off of 25kV limits the range of voltages which can be applied across the laser tube. In order to increase the acceptable voltage on the thyratron anode, a CX1826 metal/ceramic envelope thyratron is incorporated. The CX1826 is rated to 35kV. With peak and average anode current ratings of 5kA and 5A, the switch is capable

of providing the optimum voltage and current predicted by the SVL computer model of Chapter Four. Table 5.2 provides a comparison of the maximum ratings of the two thyratron types.

	CX1535	CX1826
Peak forward anode voltage (kV)	25	35
Peak forward anode current (A)	1000	5000
Average anode current (A)	1.25	5.0
Pulse repetition frequency (kHz)	100*	10

Table 5.2. Maximum permissible values of electrical parameters for the two thyratron types

* Configured in this work to run at frequencies up to this value

The CX1826[9] is a deuterium-filled, triple grid (pentode) thyratron with a ceramic/metal envelope. The greater mass of the deuterium ion over that of hydrogen results in a reduced mobility and an increase in recovery time by a factor of $2^{1/2}$ for the same tube geometry. The lower ion mobility means surface recombination effects and arc drop are reduced. The use of deuterium increases the average power capability of the CX1826; it is intended for use in metal vapour laser circuits where the capabilities of the smaller tubes are exceeded. Voltage hold-off and dissipation are improved at the expense of recovery time[13]. The CX1826 contains a dispenser cathode[14] constructed from barium aluminate impregnated tungsten.

A schematic drawing of the CX1826, which is the solid anode equivalent of the more familiar CX1625, is shown in Figure 5.4, together with the associated triggering and heater circuitry. The CX1826 has three grids. Grid 2 (G2) is the control grid. A negative bias of the order of -450V is maintained on this

grid during the OFF period to prevent the thyratron from conducting. A positive pulse of sufficient magnitude applied to G2 will trigger the thyratron. Grid 1 (G1) pre-ionises the cathode-grid region. The role of grid 3 (G3) is two-fold. Firstly, together with G1, it acts to isolate G2 from the plasma during tube recovery. The other function is to reduce the capacitance of both the cathode-grid and grid-anode regions. The connection of G3 to the cathode by means of a 10kΩ, high-wattage resistor effectively inserts an earthed electrostatic screen between G2 and the anode and reduces the grid-anode capacitance to about 0.01pF. The control grid G2 is held at a positive potential by R1 to accelerate electrons through to the anode, but effectively earthed to alternating current through a large-value capacitor.

The requirements of a modulator designed to trigger the CX1826 are to provide a DC priming current of 100mA for G1 and a voltage pulse of at least +1000V of duration 500ns followed by an inverse voltage of up to -450V to be applied to G2. The rate of rise of the G2 pulse must be a minimum of 3kV/μs. The trigger unit used is a 2kV EEV sub-modulator, capable of operation at up to 30kHz. A bias voltage of $-135V_{DC}$ is available for G1; a series-connection of two 680Ω, 17W wirewound resistors provides a priming current of 98mA. The pulse for G2 is provided by the trigger unit by switching a DC voltage, obtained via a step-up transformer and voltage doubler, with an EEV Type No. 2530 glass envelope triode thyratron across a five-section LC pulse forming network. The output pulse is obtained from a 1:1 turns ratio pulse transformer. Connection to G2 is via a 470Ω, 17W wirewound resistor. This provides a DC bias of -370V with a peak pulse forward amplitude of 3.7kV.

The CX1826 has separate reservoir and cathode heaters. As the data sheet specification[15] for both heater supply voltages is $6.3V_{AC}$, they share a common

control circuit (Figure 5.4). The replenisher is controlled by a barretter which is in turn controlled by a ballast resistor box (EEV Type No. MA942A). The two resistances are each set to 8.2Ω.

Thyratron Mounting and Cooling

Due to the high power switching capability of the CX1826 thyratron, the envelope temperature must be restricted to lie within certain limits. This is achieved by immersing the thyratron in oil, which is then circulated through a heat exchanger. The cooler oil is returned to the oil tank and directed at the base of the thyratron (cathode) to aid cooling. Immersing the thyratron in oil has the advantage of keeping the whole tube at a uniform temperature, and enables the heat to be extracted directly from the electrodes and the envelope. The oil tank is constructed from stainless steel and has dimensions 665 x 455 x 365mm (length x breadth x height). The thyratron is mounted on a support using its mounting flange. The height of the support is 100mm, which provides sufficient clearance below the mounting flange to cool the base. The trigger circuitry for the thyratron is mounted on a vertical perspex sheet adjacent to the thyratron grids. Both the heater and reservoir voltage are adjusted by means of a 240V:6.3V,10A transformer[2]. The oil tank also houses the charging inductor and diode, together with the associated protection circuitry.

5.2.1.4 Storage Capacitors

The thyratron anode is connected by means of coaxial copper wire to the storage capacitors. The storage capacitors are DHS series low-inductance, low-dissipation strontium titanate capacitors[16], rated to 40kV working voltage. Individual values range from 140pF to 2nF. In principle, the capacitor value

should be as large as possible to ensure a maximum amplitude of current pulse and hence produce a substantial Sr^{++} ion population and hence an effective recombination flux. However, restrictions are placed on the maximum value of capacitance which can be used due to the upper limit of the discharge tube temperature for lasing. As the value of capacitance is increased, in order to maintain a constant input power, the pulse repetition frequency (PRF) must be reduced. The discharge is stable down to the lowest value of PRF obtainable with the EEV sub-modulators (140Hz), and this allowed a maximum capacitance of 12nF to be used. The value of capacitance used in all the experiments described in this chapter is 8nF (four 2nF capacitors).

5.2.1.5 Circuit Layout: Grounding and Shielding

In order to minimise circuit inductance, the thyratron anode is situated within 50cm of the HV laser terminal. Connection is by means of coaxial cable rated to 50kV (between conductors) and 1kA peak current. The thyratron anode is connected in this way to the storage capacitors. The small physical size of these capacitors permits their direct mounting on the laser head by means of a 120mm wide section of copper sheet connected, together with the bypass inductor, between the cathode terminal and the coaxial return. The coaxial design of the water jacket reduces the inductance of the laser head.

Finally, the thyratron current return path to ground comprises two 100mm wide aluminium conductors mounted alongside the thyratron housing and connected to the coaxial outer braid and the thyratron support. The current return paths in the discharge circuit are connected to one common point in order to avoid ground loops. This point is chosen to be the oil tank, which is electrically connected to the earth terminal of the three-phase mains supply through the power supply.

For efficient electric field shielding, the length of the centre conductors of each coaxial cable that extend beyond the shielding braid is minimized[17].

5.2.2 LASER HEAD DESIGN

Two laser head designs are constructed. Figure 5.5 illustrates the design features of the SVL laser head designed for operation in a low heat-loss configuration. The active region is confined within a cylindrical tube constructed from Purox recrystallised alumina which is used to contain the highly reactive strontium vapour at wall temperatures of 600^{o}C-700^{o}C. Alumina has a thermal conductivity sufficiently high to remove heat from the discharge during high input power operation. Zirconia felt insulation[18] and a tantalum heat shield further reduce heat loss by convection and radiation, respectively, from the laser tube. A brass water-cooled jacket keeps the Pyrex tube cool and also acts as a coaxial return in the discharge circuit; the coaxial geometry reduces the circuit inductance.

The structure is supported by stainless-steel end-flanges (Figure 5.6). Vacuum seals near the discharge tube ends are provided by high-temperature O-rings. Barium fluoride windows (Type No. 54.113[19]) of 40mm diameter and 5mm thickness are mounted in nylon supports located at the end of each flange, inclined at an angle of 5^{o} to normal incidence to prevent the formation of a second optical cavity. The electrodes, which extend approximately 100mm into the hot region of the discharge within the aluminium tube, are made of molybdenum due to its refractory nature. The electrodes are designed to minimise aperturing of the discharge volume and establish good electrical contact between discharge circuit and electrodes. The foil electrodes (thickness 0.25mm) are rolled into cylinders of i.d. 25mm, and are mounted in

recesses in the end-flanges.

Several layers of insulation are included to reduce heat loss from the laser tube where temperatures within the range 625°C to 675°C are needed for lasing to occur (Section 4.3.4). This laser head design produced laser output which could only be supported by two 100% reflectivity mirrors. The reason for this may be two-fold. Firstly, the discharge cross-section is too large, resulting in only a small part of the cross-sectional area of the tube lying within the lasing temperature range. Secondly, only a few hundred watts of input power are required to raise the temperature of the active region to ≃625°C, with the result that, even at low pulse repetition frequencies, the discharge voltage required to maintain this temperature is below the optimum value of ≃14kV predicted by the computer analysis of Section 4.3.2.

A second design dispenses with most of the insulation and is designed to act in a high heat-loss mode, requiring in excess of one kilowatt of electrical input power. The construction is shown in Figure 5.7. The aim of this design is to fulfill the design criteria obtained from the computer model of Chapter Four for optimum recombination laser performance, ie that as high a value of stored electrical energy be discharged into the laser head as possible consistent with rapid removal of waste heat from the discharge. The new laser head consists of a 50cm length of 13mm i.d. alumina tube contained within a "bottleneck" quartz structure and supported by "dimples" in the quartz. No additional insulation is present other than approximately 1cm lengths of Saffil[20] alumina fibres wrapped around each end of the alumina tube to prevent the discharge entering the alumina - quartz region. This permits the attainment of higher average input powers than can be achieved with the previous design whilst maintaining the same strontium temperature. To compensate for the additional heat loss, the PRF is

increased from 800Hz to 3kHz. A fan is employed to reduce the temperature of the discharge volume.

Serrated edges on the electrode ends help to stabilise and initiate the discharge. The electrode separation is 45cm, giving an extraction volume of $58cm^3$. The optical resonator consists of a one metre radius of curvature high reflector at 430.5nm[21] at the anode end and a six metre radius of curvature (reflectivity 47% at 430.5nm) output coupler at the cathode end. This reflection coefficient is within the optimum range of 30-50% for maximum extracted energy reported by Butler and Piper[22]. The cavity length is two metres.

5.3 DIAGNOSTICS

The following parameters are monitored. These topics are discussed in the order listed below.

1. Laser cathode and thyratron anode voltages
2. Laser and bypass inductor currents
3. Laser tube temperature
4. Spontaneous emission
5. Average laser power
6. Laser pulse duration

All voltage, current and laser pulse waveforms are displayed on a Hewlett-Packard digital oscilloscope (Hewlett-Packard Type No. 54112D)[23], which allows simultaneous display of up to four waveforms. The coaxial cables running from the voltage and current probes and the photodiode to the oscilloscope are

wrapped around ferrite cores to reduce the amount of noise entering the oscilloscope.

5.3.1 VOLTAGE DIAGNOSTICS

All thyratron grid and anode, storage capacitor and laser cathode voltage waveforms are monitored by means of a Tektronix Type No. P6015 high voltage probe[24]. This attenuates the voltage measured by a factor of 1000x and has a specified rise-time of 4ns when terminated in a 1MΩ impedance. In order to measure rapidly varying voltages, the connections between the voltage probe and the points of measurement are designed to minimise inductance, since a voltage may be induced by means of Ldi/dt.

5.3.2 CURRENT DIAGNOSTICS

The most important parameter measured in a pulsed circuit is the current, i(t). This is particularly true for recombination lasers where precise tailoring of the discharge current is essential. Consideration of Figure 5.1 reveals that the current waveforms of interest are those through the laser head, bypass inductor and thyratron. Pulsed currents are measured by means of an Ion Physics Type No. 0603 current loop[25]. This device has a resolution of 10A/V. A 50Ω BNC termination is used to match the current probe impedance to the oscilloscope, the rise-time of which is better than 0.1ns. The noise level produced is negligible.

5.3.3 TEMPERATURE MEASUREMENTS

Temperature measurements are made internal to the discharge tube by means of

a Minolta/Land Cyclops Type No. 52 infra-red digital optical pyrometer[26]. This provides accurate readings in the temperature range 600°C - 3000°C, and is consequently suitable for the monitoring of the temperature within the laser tube. The emissivity selector on the optical pyrometer is set at 0.4 for alumina. The pyrometer is calibrated against the melting point of copper in a copper vapour laser[27]. Temperature readings within the discharge are taken by focusing the instrument down to a 5mm diameter spot (or less) through the discharge onto the tube wall. An average of three readings are taken for each measured temperature. This technique has the disadvantage of focusing through the discharge. Account is taken in the interpretation of the readings of the fact that the on-axis temperature may vary significantly from that at the wall.

5.3.4 SPONTANEOUS EMISSION

In order to maximise the spontaneous emission at the laser wavelength in seeking to determine the conditions which may lead to lasing, the light from the discharge tube is passed through a spectrometer tuned to 430.5nm and onto the window of a UV photomultiplier tube. The photomultiplier tube is biased at -20kV in order to maximise tube gain. The output of the photomultiplier is terminated in 50Ω and fed to a chart recorder. To prevent saturation of the photomultiplier tube, neutral density filters are placed before the spectrometer and the spontaneous emission is split using a prism.

5.3.5 AVERAGE LASER POWER

Average laser power is measured by means of a Scientech model 362 power/energy meter[28]. Since this is a thermopile detector, the power meter is placed as far away from the laser head as is practicable (approximately three

metres) in order to avoid any heating effects caused by thermal radiation emitted by the discharge through the laser window. Background values are deducted from actual readings in order that the true laser power is recorded.

5.3.6 LASER PULSE SHAPE

The intensity and duration of the stimulated emission pulses are observed using a photodiode[29]. Neutral density filters are placed before the photodiode in order to prevent saturation of the photodiode and consequent distortion of the pulse shape. The photodiode is positioned to monitor the central portion of the discharge. The photodiode is placed inside a metal box in order to screen out rf noise generated by the laser and pulsed power circuitry.

5.4 EXPERIMENTAL METHOD

5.4.1 STRONTIUM STORAGE AND HANDLING

Pure strontium metal has a melting point of 768^{o}C[30]. However, strontium reacts spontaneously with water, or even moist air, liberating hydrogen and heat. The possibilities of explosion and fire thus present hazards to the user, especially to the hands and face. Even on more gradual exposure to air, strontium readily oxidises. Pure strontium oxide has a melting point of 1600^{o}C so oxidation must be prevented.

Strontium metal (purity >99%) is available in the form of a 15mm diameter rod[30], stored under a hydrocarbon liquid or ampouled under an inert gas such as argon in a glass container, in order to prevent oxidation. The strontium used in these experiments is stored under liquid paraffin which has a high boiling

point. Small pieces of the metal, of dimensions typically 2 or 3mm, are cut from this rod with a scalpel whilst immersed in the paraffin. In order to maintain an inert atmosphere during the loading of the strontium into the discharge tube, the window at the anode end of the laser head is removed and helium is flowed through the active region from the cathode end. The strontium pieces are removed from the paraffin and immersed in petroleum (of lower melting point than liquid paraffin at 60-80^{o}C) in order to displace it. They are introduced through the anode electrode in groups of five or six and positioned along the bottom of the alumina tube at intervals of approximately 9cm, whilst still soaked in the protective spirit. Petroleum is a light hydrocarbon that has an evaporation time of a few minutes at room temperature. It is found that removing the strontium from storage under liquid paraffin and scraping the metal clean with a scalpel prior to loading in the tube is satisfactory. The tube is then sealed by replacing the anode window and completing the vacuum seals. The helium buffer gas flow rate is reduced and the source is pumped down, as usual.

5.4.2 PASSIVATION OF THE DISCHARGE TUBE

The pressure of helium within the discharge tube is increased to 25mbar. The tube is operated under self-heating conditions. A discharge is excited in the tube by means of the pulse generator circuit of Figure 5.1. The circuit produces voltage pulses with an amplitude up to 40kV. Peak currents up to 1kA can be achieved in discharge pulses of duration typically 200ns (FWHM). A PRF of 10kHz can be achieved.

For a given storage capacitance, optimum matching of the discharge circuit impedance to that of the discharge (and consequently minimum current ringing) is obtained for a given combination of gas pressure and charging voltage. The PRF

is set at 1kHz. At initial turn-on, the impedance of the plasma is high and the matching is poor (Figure 5.8); the FWHM of the current pulse is ~1μs. As the voltage across the discharge tube is increased, the plasma temperature rises and, as strontium vapour is released, the plasma impedance falls, which produces better matching (Figure 5.8). However, as the temperature increases further, contaminants present in the alumina and gas supply tubes are released into the discharge which serves to counteract the fall in impedance. This outgasing can prevent laser action. The discharge voltage is increased until the inner wall of the discharge tube is at approximately the correct temperature for lasing, monitored by focusing the optical pyrometer through the discharge. The current pulse through the laser head is monitored. The discharge is terminated and the laser tube is evacuated. The tube is re-filled with helium. This process is repeated until the current pulse exhibits minimal ringing. The value of bypass inductor is now adjusted to further prevent significant current ringing and ensure rapid termination of the current pulse at the lasing temperature. The value of bypass inductor chosen is 110 μH (this value allows the thyratron to recover). A discharge current pulse of duration ~200ns (FWHM) was measured with an ITT Rogowski loop[25] (Figure 5.8).

5.4.3 SPONTANEOUS EMISSION

The characteristic discharge colour of strontium is light blue. As the strontium melts onto the tube wall, the absence of extra thermal insulation allows the state of each cluster of strontium to be viewed as it melts. Once all of the six clusters have melted, the discharge voltage and PRF are adjusted to maximise emission on the laser transition. Maximum emission is obtained at a discharge voltage of 14.4kV and a prf of 800Hz. The inner tube wall temperature is 630 ± 10^{o}C. Between one and two hours passivation at this temperature are

necessary before spontaneous emission on the relevant SrII transition maximises and laser emission can be obtained. This is the subject of the following section.

5.5 EXPERIMENTAL RESULTS

5.5.1 GENERAL CHARACTERISTICS

A static fill of helium at 200 mbar is present within the discharge volume. With the circuit parameters detailed in Section 5.2, the addition of a 47% reflectivity output coupler produced the stimulated emission pulse shown in Figure 5.9. The laser cathode voltage and discharge current pulse are shown in Figure 5.10 for reference. The stimulated pulse duration, τ_p, is ~350ns (FWHM). The recombination behaviour can be seen clearly. The long-lived tail - extending beyond one microsecond - of the recombination peak is predicted from the state variable analysis (Figure 4.16). It is important to note that, as predicted by the model, laser emission occurs in a single intense pulse and commences as the magnitude of the discharge current tends to zero (≤10% of the peak amplitude). This result confirms that, for efficient recombination pumping, a zero applied field is necessary for rapid plasma cooling after cessation of the current pulse. No stimulated emission is observed in the presence of significant current ringing (>20%) or for cases in which the current is not rapidly terminated.

Although the laser can be operated with static gas fills, helium is usually flowed through the discharge tube at ~1$cm^3 min^{-1}$ (at atmospheric pressure) to remove residual contaminants as passivation proceeds.

Table 5.3 provides a comparison of the experimentally observed characteristics of the laser cathode voltage, discharge current and stimulated emission pulses at 630°C with the results of the state variable analysis of Section 4.3.4 for cavity temperatures of 625°C and 675°C. Good agreement is obtained for all parameters except for the duration of the stimulated emission pulse.

	Value			
	State Variable Analysis		Experiment	Units
Optimum temperature of discharge tube wall	625	675	630	°C
Optimum charging voltage for lasing	14.4	14.4	14.4	kV
Amplitude of discharge current pulse	85	183	150	A
Duration of discharge current pulse (FWHM)	350	250	350	ns
Delay between peak of current and stimulated emission pulses	500	450	450	ns
Duration of stimulated emission pulse (FWHM)	250	170	350	ns

Table 5.3. A comparison of experimentally observed values of the main discharge parameters with computer model predictions

Output power showed no degradation to within ±10% over a three hour period of operation, indicating that passivation of the full area of the alumina tube bounding the active volume is complete. An average power of ~0.3W is reliably achieved. This corresponds to a peak power given by

$$\text{Peak power} = \frac{\text{Average power}}{\tau_p(\text{FWHM}) \times \text{prf}} = 1.05 \text{ kW}. \qquad (5.20)$$

The laser output energy per pulse is ≃375μJ and the output energy per unit volume is estimated to be 8.1 μJcm^{-3}, assuming that 80% of the laser tube volume lies within the temperature range for lasing. The cylindrical electrode design ensures that there is insignificant aperturing of the discharge cross-section.

Speckle is present in the laser spot. Although no attempt is made to measure the spatial and temporal coherence of the beam, these are expected to be small due to the high gain and short pulse lengths (~350ns (FWHM)) of the emission. The laser output intensity appears to be quite uniform in profile and no mode structure is observed. The beam divergence measured is less than 10mrad.

Maximum laser power is a function of helium pressure, temperature of the active medium (and hence strontium vapour pressure), and amplitude and rate of decay of the discharge current pulse. The dependence of laser output power on each of these parameters is described below. Observations of output powers are made using an output coupler of reflectivity 47%. Based on the observations of Butler and Piper[22], optimum output power is observed for a reflectivity of the output coupler within the range 30-50%. Despite the relatively high gain of the SVL ($4.3\%cm^{-1}$, Section 4.3.1), no superradiance is observed on removing the output coupler.

5.5.2 EFFECT OF HELIUM PRESSURE ON THE CHARACTERISTICS OF THE DISCHARGE CURRENT PULSE

In principle, the SVL should be run at as high a buffer gas pressure as possible since this increases the plasma impedance - leading to greater energy deposition in the gas - and lowers the electron temperature more rapidly in the discharge current afterglow. Figure 5.11 shows the current pulses through the

discharge tube at various helium pressures. This illustrates that the discharge current pulse amplitude increases and the fall-time decreases as the helium pressure is decreased.

5.5.3 EFFECT OF TEMPERATURE OF THE ACTIVE MEDIUM ON AVERAGE LASER OUTPUT POWER

The SVL computer code predicts that the optimum small-signal gain coefficient is achieved for a storage capacitor charging voltage of 14.4kV at a temperature of the active medium within the range 625^{o}C to 675^{o}C. By maintaining a constant storage capacitance, the temperature within the laser tube is varied by means of the discharge voltage and PRF. The tube temperature is observed with the optical pyrometer (Section 5.3.3).

During the measurements, the voltage on the storage capacitors is adjusted at each value of PRF to produce the maximum laser output power. Figure 5.12 illustrates the dependance of laser output power on PRF of the excitation circuit (equivalent to a range of tube temperatures). Maximum laser power is achieved at a PRF of 3kHz. Lasing occurs in the temperature range 590-630^{o}C ($\pm$20^{o}C). This corresponds to an optimal strontium vapour pressure of ~0.006-0.015mbar[31]. Under these conditions, ouput energies of 170μJ per pulse are obtained in 250ns (FWHM). The peak laser power and specific energy density are calculated to be 0.5kW and 4μJcm^{-3}. Laser efficiency, which is defined as the percentage of electrical energy stored in the capacitors which is converted to optical energy, is $\simeq$0.05%. This value compares with a value of 0.10-0.14% obtained by Zhukov[32].

5.5.4 EFFECT OF AMPLITUDE OF APPLIED EXCITATION PULSE ON AVERAGE LASER OUTPUT POWER

The results of the computer modelling described in Chapter Four indicate that optimum laser output power is achieved by the application of a high amplitude, well-terminated current pulse producing a cavity temperature within the range 625°C to 675°C. The amplitude of the current pulse is adjusted by means of the capacitor charging voltage. The buffer gas pressure is maintained constant at 110 mbar. Figure 5.13 shows the variation of laser output power with amplitude of the discharge current pulse for various values of the PRF. Figure 5.13 indicates that as the PRF increases, the optimum amplitude of the current pulse for lasing also increases. The highest average power is obtained for a current pulse magnitude of 220A (current density 165Acm^{-2}) and duration ~180ns (FWHM) at a pulse repetition frequency of 3kHz.

5.6 CONCLUSIONS

Stimulated emission on the 430.5nm $6^2S_{1/2}$-$5^2P_{3/2}$ transition of strontium II has been demonstrated. The strontium vapour laser is operated under self-heating conditions in a high heat loss configuration and reliably produces 300mW (96% output coupling) for input powers of 1kW. The laser output power shows no degradation in magnitude to within ± 10% after several hours of operation under sealed-off conditions. Lasing is observed only for a narrow range of input energies which translates to a strong dependence of laser output power on gas temperature. Rapid relaxation of the plasma between pulses is required to reduce the electron and buffer gas temperatures simultaneously.

The computer model results of Chapter Four agree well with experimental

observations of cathode voltage and discharge current amplitudes and durations. Further agreement is achieved for the time of initiation and characteristic shape of the stimulated emission pulse and for the range of cavity temperatures for which lasing is observed.

A strong dependence of the laser output power on the characteristics of the excitation pulse is exhibited. The circuit described in this chapter is capable of generating 450 Acm^{-2} in current pulses of duration ~180ns (FWHM). Improvements in laser performance are anticipated if the fall-time of the applied current pulse can be decreased. A method of rapidly terminating the current pulse applied to the SVL is investigated in the following chapter.

REFERENCES FOR CHAPTER 5

1. GD Rectifiers Ltd., 4, Victoria Gardens, Burgess Hill, West Sussex, England. RH15 9NB.

2. TEC (Transformer and Electrical Company), Honywood Road, Basildon, Essex, England. SS14 3DT.

3. G. N. Glasoe and J. V. Lebacqz, 'Pulse Generators', MIT Radiation Laboratory Series No.5, McGraw-Hill Book Co. (1948).

4. Electronic Devices, Inc., 21, Gray Oaks Avenue, Yonkers, NY 10710.

5. F. Goodenough, "Power FETs switch 25kW in under 100ns", Electronic Design, 34 (23), pp.41-42, Oct 2, 1986.

6. R. M. Ness, E. Y. Chu and G. T. Santamaria, "0.5MW 60kHz solid state power modulator", Eighteenth Power Modulator Symposium, pp.43-47, Hilton Head, SC, June 20-22, 1988.

7. Brown, Boveri & Cie, 'Brown Boveri Power Semiconductors Catalogue', Aktiengesellschaft, Bereich Halbleiter, Postfach 1180, D-6840 Lampertheim, Switzerland.

8. J. Vitins, J. L. Steiner and A. Schweizer, "Reverse conducting thyristors replace thyratrons in sub-microsecond pulse generation", Sixth IEEE Pulsed Power Conference, pp.591-594, Arlington, VA, June 29-July 1, 1987.

9. P. D. Culling, H. Menown, C. A. Pirrie and C. A. Roberts, "Recent developments in high repetition rate thyratrons for copper vapour lasers", Sixth IEEE Pulsed Power Conference, pp.598-603, Arlington, VA, June 29-July 1, 1987.

10. P. Billault, H. Riege, M. van Gulik, E. Boggasch, K. Frank and R. Seebock, 'Pseudospark Switches', CERN Report No. 87-13 (1987).

11. C. A. Pirrie, "Recent advances in thyratron design", IEE Colloquium on Pulsed Power Technology, Savoy Place, London, Dec 8-9, 1988, Digest No. 1988/133.

12. H. Menown, C. A. Pirrie and N. S. Nicholls, "Advanced thyratrons as switches for the nineties", IEEE Seventeenth Power Modulator Symposium, Seattle, WA, pp.69-73, June, 1986.

13. H. N. Price, "Deuterium versus hydrogen in thyratron modulator switch tubes", Eighth Symposium on Hydrogen Thyratrons and Modulators, pp.18-30, Fort Monmouth, NJ, May 12-14, 1964.

14. A. W. Hull, "The dispenser cathode - a new type of thermionic cathode for gaseous discharge tubes", Phys. Rev., 56, pp.88-93 (1939).

15. 'Hydrogen Thyratrons Products Data', Reference Manual, EEV Co., Ltd., Waterhouse Lane, Chelmsford, Essex, England. CM1 2QU. (June, 1978).

16. Murata Mfg. Co., Ltd., Nagaokakyo-shi, Kyoto 617, Japan.

17. H. W. Ott, 'Noise Reduction Techniques in Electronic Systems', Wiley-Interscience (1976).

18. Zircar Products, Inc., 110, North Main Street, Florida, NY 10921.

19. Analytical Accessories Ltd., Unit 3, Lagoon Road, St. Mary Cray, Orpington, England. BR5 3QX.

20. ICI Chemicals and Polymers Group, PO Box No.14, The Heath, Runcorn, Cheshire, England. WA7 4QG.

21. Technical Optics, Second Avenue, Onchan, Isle of Man, England.

22. M. S. Butler and J. A. Piper, "Optimization of the excitation channels in the discharge-excited Sr^+ recombination laser", Appl. Phys. Lett., 45 (7), pp.707-709 (1984).

23. Hewlett-Packard, Ltd., 1/3 Springburn Place, College Milton North, East Kilbride, Scotland. G74 SN4.

24. Tektronix HV probe Type No. P6015, Tektronix, Inc., Beaverton, OR.

25. Ion Physics Company, PO Box 416 B, MS 0603.

26. Land Instruments, Inc., PO Box 1623, Tullytown, PA 19007.

27. G. L. Clark, 'Investigation of Copper and Gold Vapour Lasers', Ph.D. Thesis, University of St. Andrews (1988).

28. Photon Control, Ltd., Kings Court, Kirkwood Road, Cambridge, England. CB4 2PF.

29. ITL sub-nanosecond photodiode, Type No. TF1850M20 operating instructions.

30. 'Metals and Materials for Research and Industry', Catalogue, Goodfellow Metals, Ltd., Cambridge Science Park, Milton Road, Cambridge, England. CB4 4DJ (1990).

31. A. N. Nesmeyanov, 'Vapour Pressure of the Elements', pp.185-187, MacMillan and Co. Ltd. (1963).

32. V. V. Zhukov, V. S. Kucherov, E. L. Latush and M. F. Sèm, "Recombination lasers utilizing vapours of chemical elements. II. Laser action due to transitions in metal ions", Sov. J. Quantum Electron., 7, No.6, p.708 (1977).

FIGURES FOR CHAPTER 5

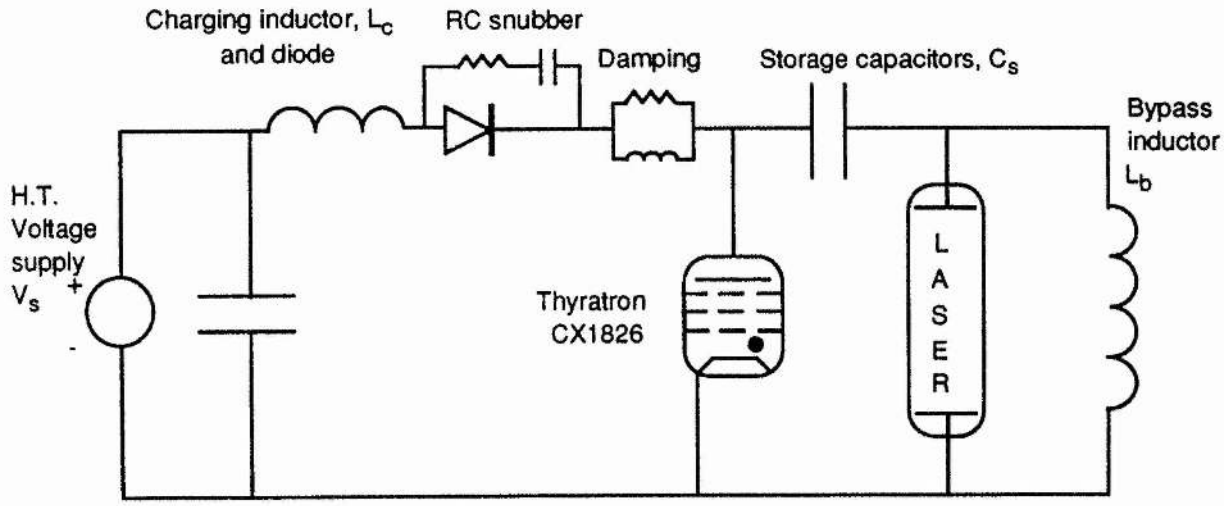


Figure 5.1. Electrical Discharge Circuit

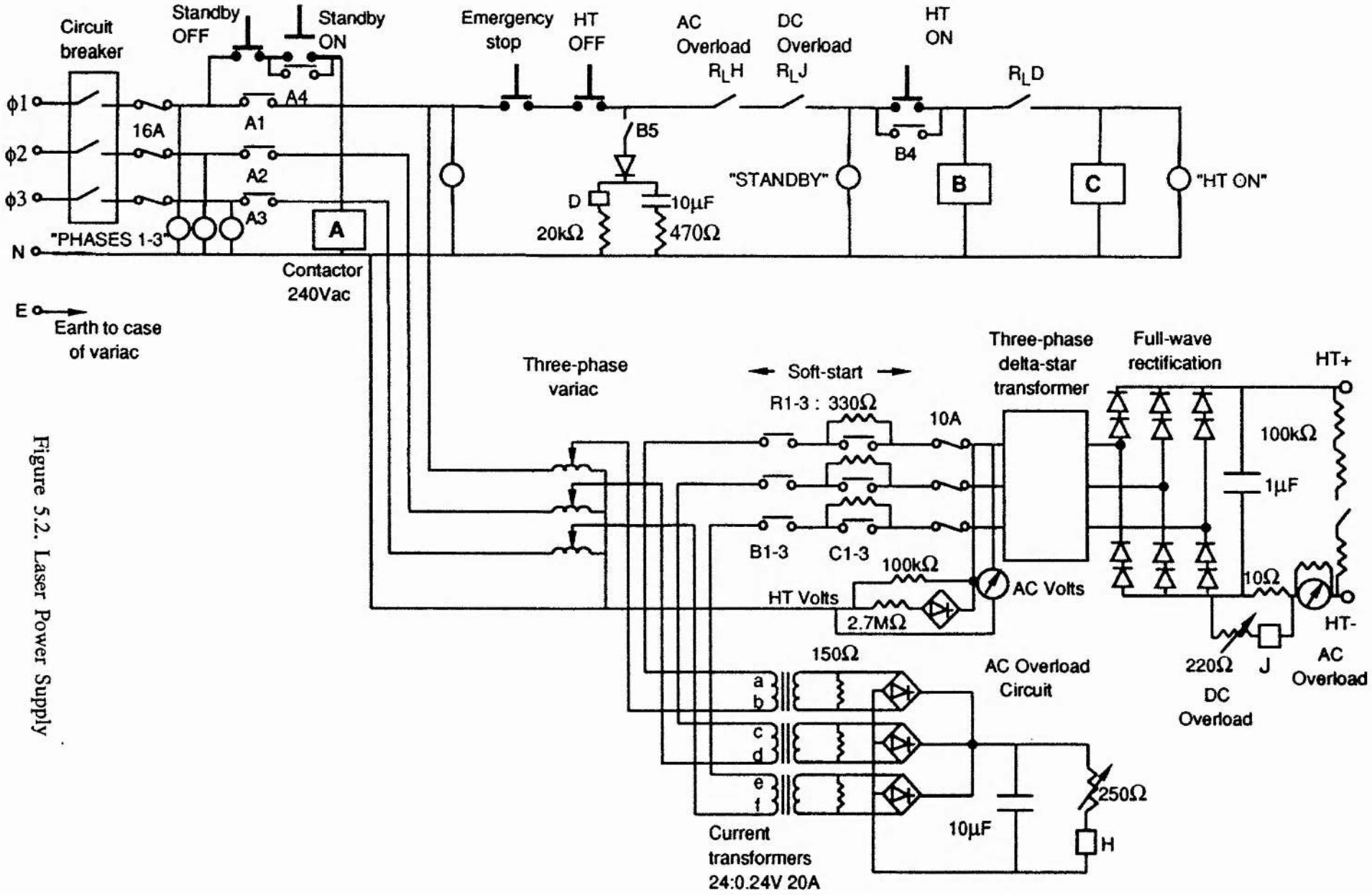


Figure 5.2. Laser Power Supply

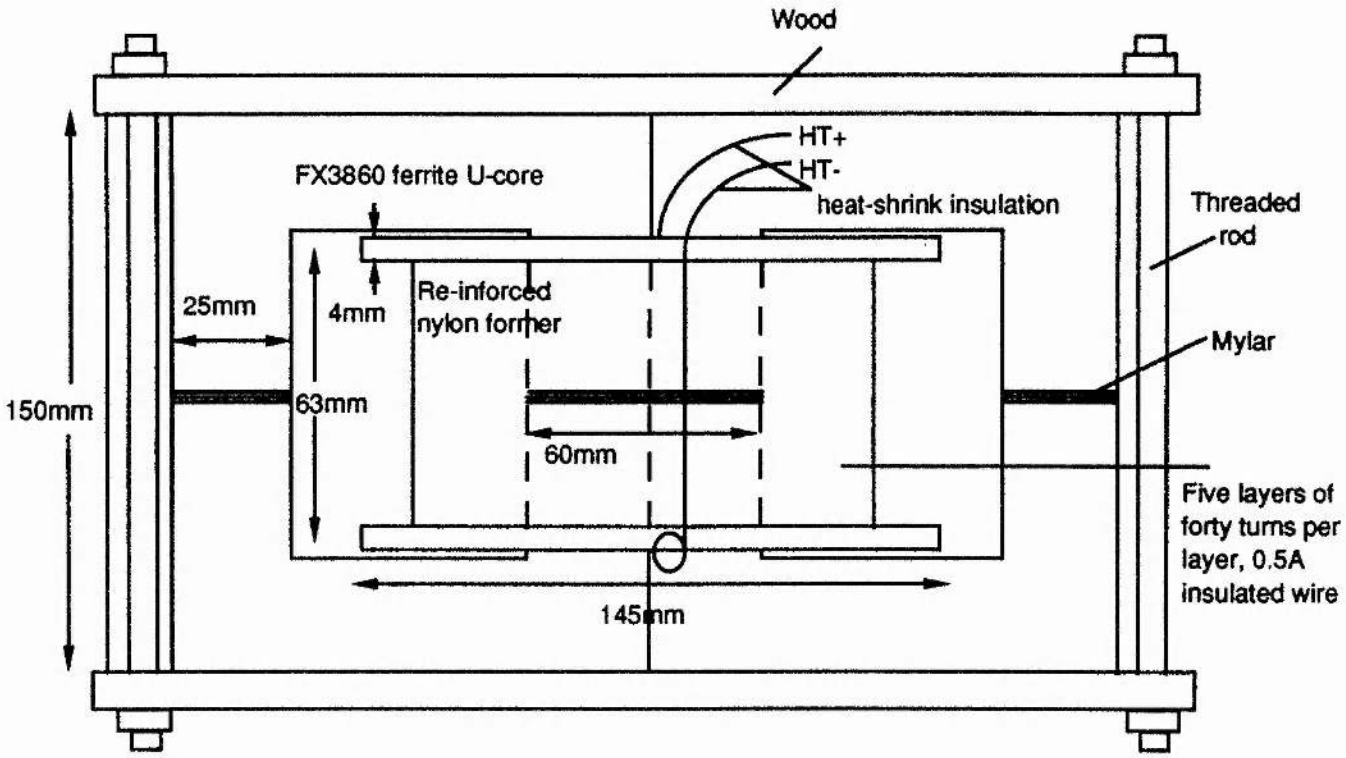


Figure 5.3. Construction of resonant charging inductor

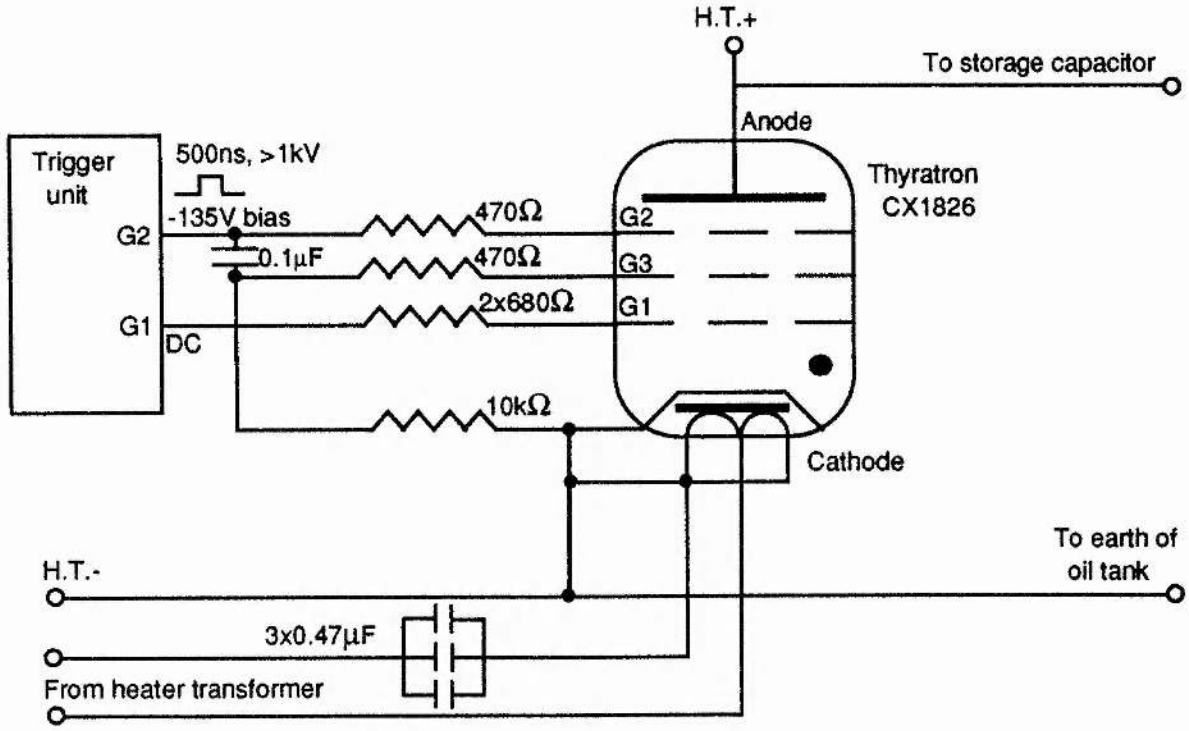


Figure 5.4. Outline drawing of CX1826 thyratron including triggering and cathode and reservoir heater circuits

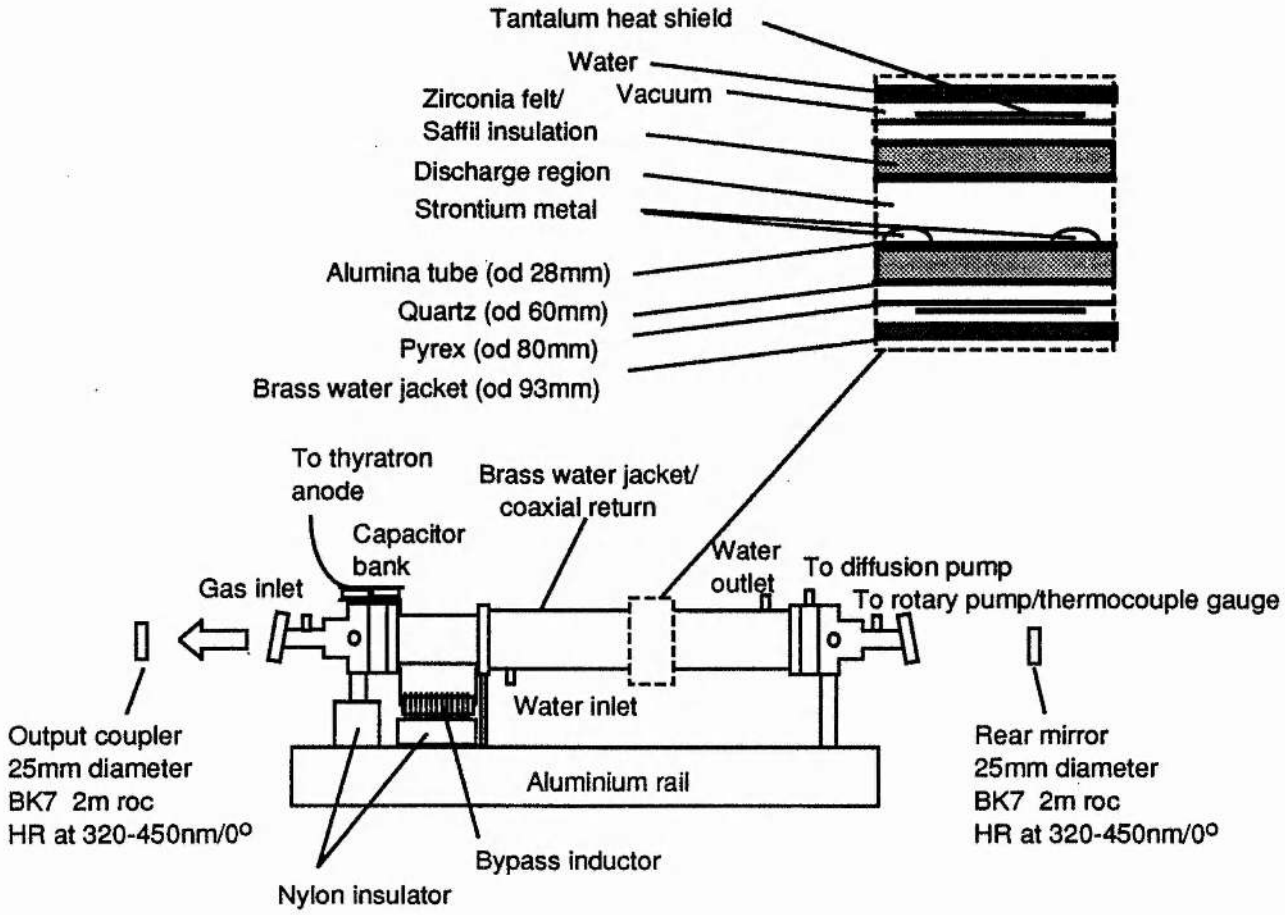


Figure 5.5. Laser head design for the low heat-loss mode of operation

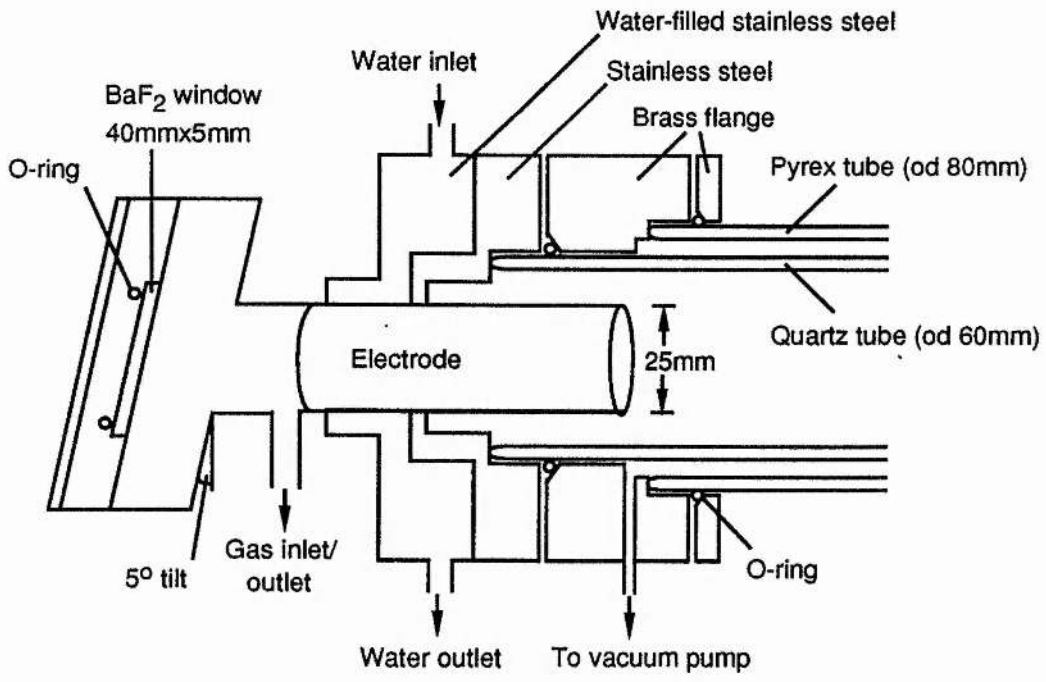


Figure 5.6. End-flange construction

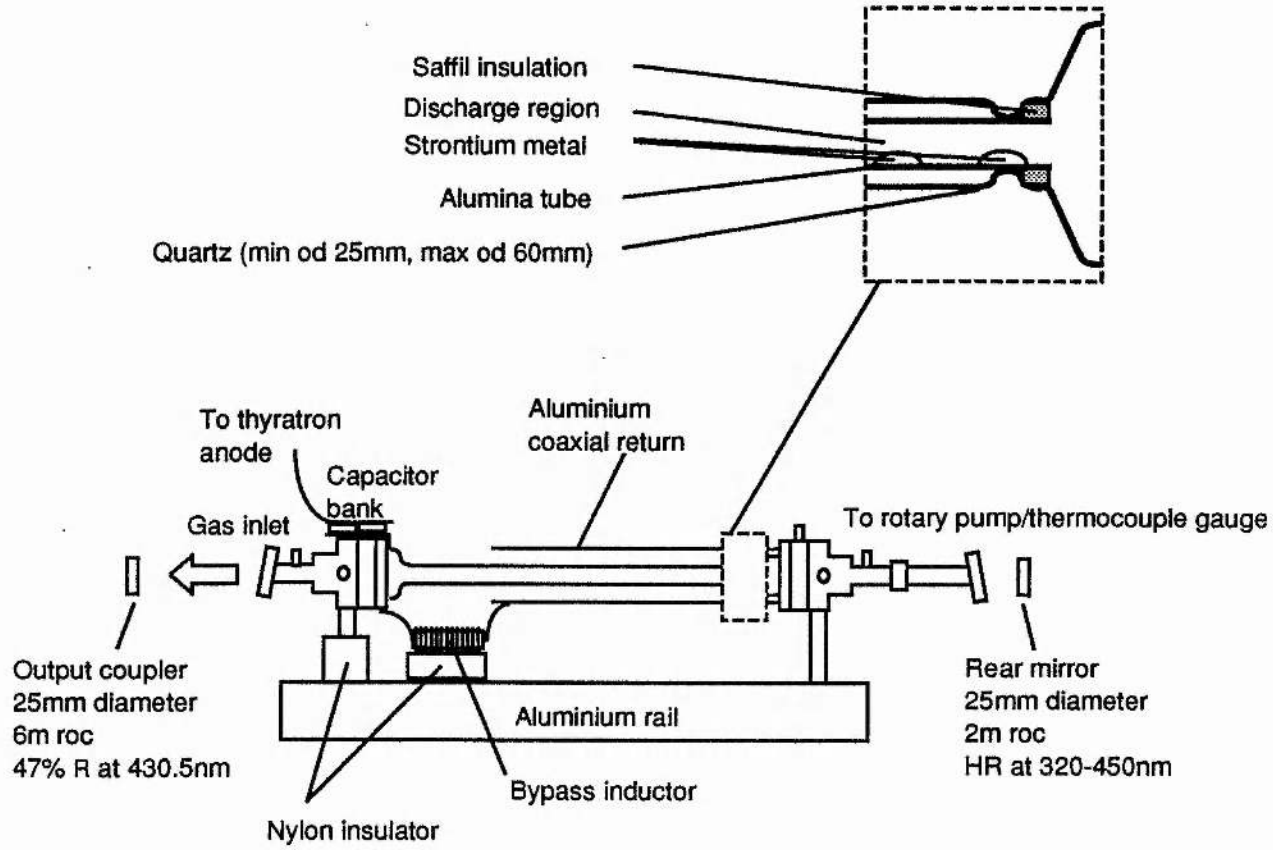


Figure 5.7. Laser head design for the high heat-loss mode of operation

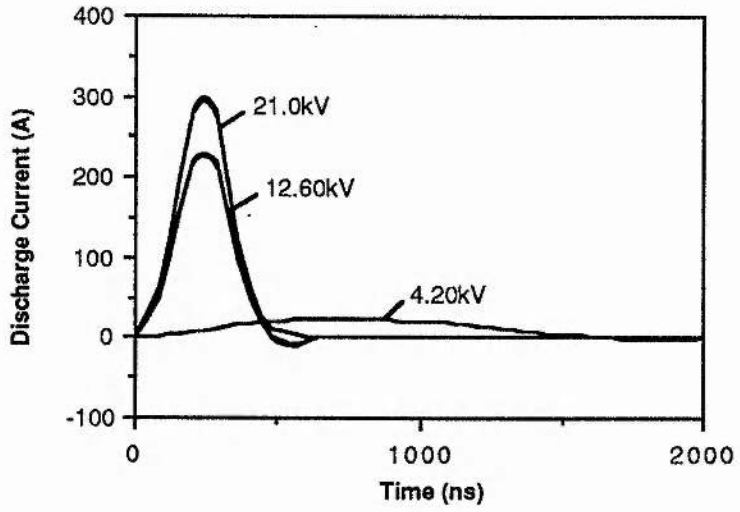


Figure 5.8. Impedance matching of the excitation circuit and discharge load (a) V_{Cs} = 4.2kV, p_{He} = 30mbar (b) V_{Cs} = 12.6kV, p_{He} = 30mbar and (c) V_{Cs} = 21.0kV, p_{He} = 50mbar

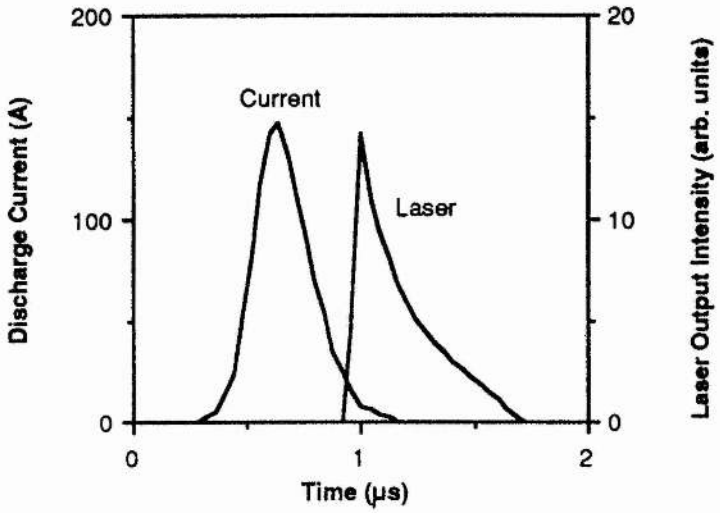


Figure 5.9. Experimental discharge current and stimulated emission pulse

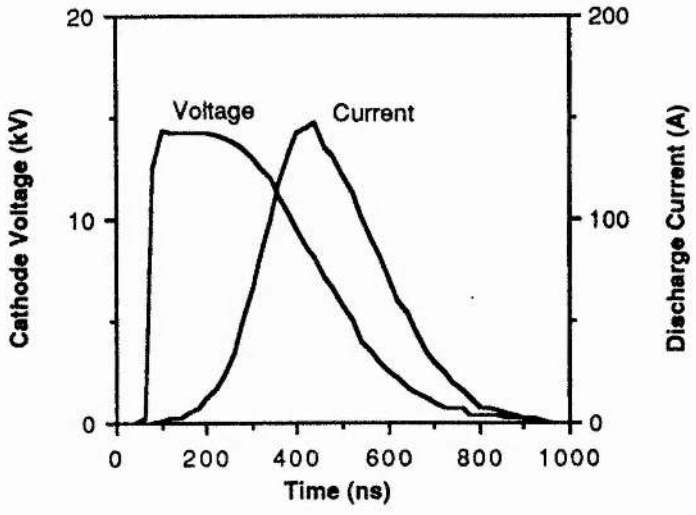


Figure 5.10. Experimental laser cathode voltage and discharge current

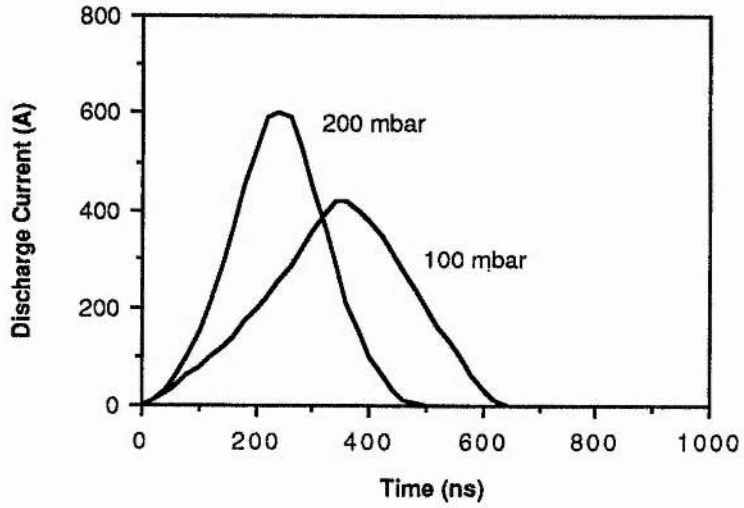


Figure 5.11. Discharge current pulse waveforms produced in helium buffer gas at a pressure of (a) 100 mbar and (b) 200 mbar. The pulse repetition frequency is 3 kHz

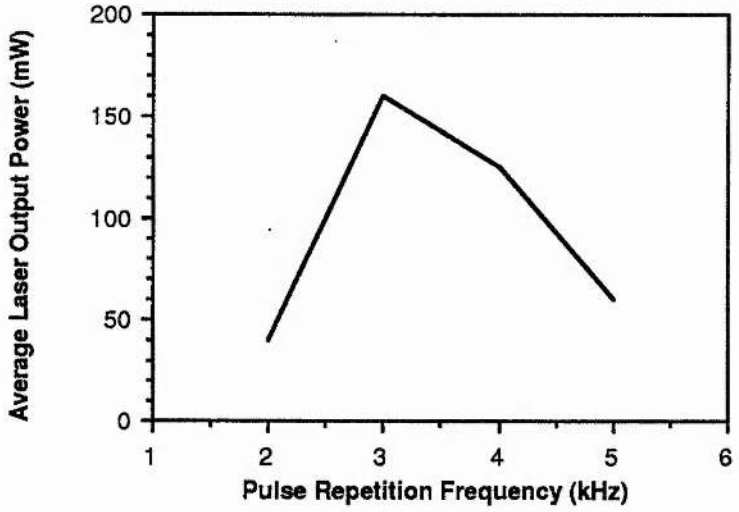


Figure 5.12. Dependence of average laser output power on the pulse repetition frequency in helium at a pressure of 110 mbar. V_{Cs} = 22.7kV

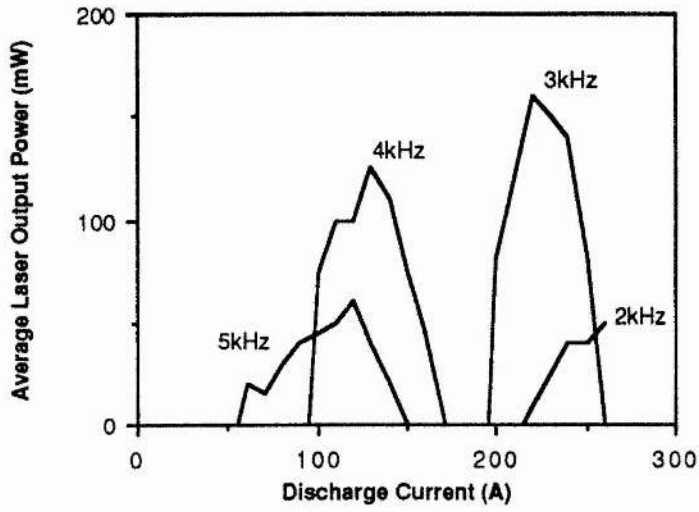


Figure 5.13. Dependence of average laser output power on the amplitude of the discharge current pulse in helium at a helium pressure of 110 mbar and a pulse repetition frequency of (a) 2kHz, (b) 3kHz, (c) 4kHz, (d) 5kHz

Chapter 6

A Method of Rapidly Terminating the Current Pulses Applied to Recombination Lasers

6.1 INTRODUCTION

The results of Section 4.3 indicate that the key to volume scaling of the strontium vapour laser is the simultaneous achievement of low gas temperature and low electron temperature across the full diameter of the discharge tube. Laser output is maximal when the discharge current pulse falls off steeply. This produces rapid relaxation of the plasma between pulses, enabling recombination to proceed. The computer analysis of Section 4.3.4 reveals that for a 13mm id discharge tube containing helium buffer gas at pressures above 200mbar, the optimum voltage applied to the laser head should be 14.4kV, producing a discharge current of approximately 180A. The rise-time of the current pulse is satisfactory at 100-200ns, but the current pulse should be terminated as rapidly as possible. Figure 6.1 shows the laser cathode voltage and discharge current and bypass inductor current waveforms obtained operating the C-to-C transfer circuit of Figure 5.4. This circuit uses an air-cored inductor as the bypass element. The laser current fall-time of Figure 6.1a is approximately 275ns (90%-10%) which corresponds to a rate of fall of current considerably slower than the optimum value of 50ns.

This chapter describes a circuit capable of generating current pulses with peak amplitudes up to 300A (400 Acm^{-2}) and fall-times of less than 70ns (90%-10%). The system consists of a thyratron-switched capacitive discharge circuit in which the bypass element is a saturable inductor. Saturation of the

magnetic core provides an alternative path for the discharge current. Furthermore, the point in time at which saturation occurs can be controlled by appropriate magnetic core biasing. This enables the current pulse to be terminated at its peak amplitude and in principle alter its magnitude.

As the work described in this chapter incorporates the design of linear and saturable inductors for the modulator circuit, a basic review of the fundamental magnetic design principles is given. This includes a discussion of magnetic core materials and configurations available. The principles of operation underlying the technique of magnetic pulse sharpening, are explained. In contrast to the triggered switches described in Chapter Five, magnetic switching is passive and depends upon the non-linear behaviour of a magnetic material when subjected to sufficiently strong flux density. The switching action is achieved by the existence of two distinct values of inductance depending on whether the core is saturated (switch OPEN) or unsaturated (switch CLOSED). An outline of the design parameters relevant to saturation is included.

A flux-controlled model of the saturable bypass inductor is described in Section 6.4.2. A state variable analysis is performed in order to simulate the performance of the laser driven by this modulator and the results are validated by experiment.

6.2 MAGNETIC CIRCUIT REVIEW

A magnetic material offers an opposition to the establishment of a magnetic flux ϕ set up by a current flowing in a surrounding conductor. This opposition is termed the reluctance R and is given by

$$R = \frac{l_e}{\mu_e A_e} \tag{6.1}$$

where l_e (cm), A_e (cm^2) and μ_e are the effective magnetic path length, cross-sectional area and permeability of the core. The magnetomotive force F thus produced is given by

$$F = \phi R \tag{6.2}$$

When the reluctance drop is distributed along the length of the magnetic path, the magnetising force that produces the magnetic flux is the magnetic field strength, or magnetising force, H and is derived from Ampère's circuital law

$$\int \underline{H}.\underline{dl} = 0.4\ \pi\ NI \tag{6.3}$$

where N is the number of turns of conductor and I the current.

Hence, we get

$$H = \frac{0.4\ \pi\ NI}{l_e} \quad \text{Oersteds.} \tag{6.4}$$

Combination of equations (6.1), (6.2) and (6.4) yields

$$\phi = \frac{0.4\ \pi\ NI\ \mu_e}{l_e} . A_e$$

$$= B\ A_e \tag{6.5}$$

where B is the magnetic induction (in gauss) defined by

$$B = \mu_e H \quad . \tag{6.6}$$

6.2.1 MAGNETIC MATERIALS

Core materials can be broadly grouped into four categories, the characteristics of which are listed in Table 6.1. Laminations consisting of transformer sheets are used in filters. Powder cores consist of electrically insulated grains of iron powder mixed with a binding medium which finds applications in high Q filters and rf transformers.

Core Type	Core Material	Flux Density	Frequency of Operation
Laminations	Si-Fe		1-100Hz
	Ni-Fe	High	40-100kHz
	Co-Fe		1-5kHz
Molybdenum Permalloy Powder (MPP) and Iron Powder Cores	80% Nickel	Medium	DC
Ferrites	FeO+ NiO + MnO + ZnO	Low	MHz 2kHz-200kHz

Table 6.1. A comparison of the properties of magnetic core materials

Soft ferrites are dense ceramic structures of ferromagnetic oxides mixed with one or several oxides of metals such as nickel, manganese or zinc and are the only magnetic materials used in power electronics designs. Manganese oxide ferrites find application in the 2kHz to 200kHz range. Despite its modest operating flux density - B_{sat} lies between 3,000 and 5,000 gauss - it does offer

low core losses at high frequencies, good winding coupling and ease of assembly due to the wide range of core geometries available.

In this thesis, the ferrite used is predominantly 3C8 material, manufactured by Mullard as part of the Ferroxcube series[1]. This is a Mn-Zn ferrite suitable for power applications with a relatively high operating flux density, low total core loss and an operating frequency up to 300kHz. This material is used in the design of an inductor for resonant charging of the pulse-forming network in the laser modulator circuit. However, for the saturable bypass inductor , two further ferrite materials are investigated; these are L4A (nickel-zinc) and lithium titanium zinc[2]. A comparison of the three ferrite materials is presented in Table 6.2.

Core Material	ΔB_{max}(T) at 25°C	μ_{unsat}	μ_{sat}
Mn-Zn (3C8)	> 0.90	2000	< 10
Ni-Zn	0.80	500-2000	< 5
Li-Ti-Zn	0.65	1000-2000	< 5

Table 6.2. Parameters for the ferrite materials employed

Magnetic materials are characterised by a plot of the magnetic induction as a function of the magnetising force. The hysteresis loop for 3C8 material at 25°C is shown in Figure 6.2. As the frequency of the applied voltage is increased (assuming constant flux density), the width of the loop increases due to eddy current losses in the core. The maximum excursion in magnetic induction ΔB_{max} to maintain linear operation is twice the saturation value B_{sat}.

6.2.2 INDUCTOR DESIGN CRITERIA

When the winding of a magnetic component is excited by a flux ϕ, the instantaneous voltage induced v(t) is defined from Faraday's law either in terms of the rate of change of current through it or the rate of change of flux in its core:

$$v(t) = L \frac{di}{dt} = KN \frac{d\phi}{dt} \tag{6.7}$$

where K is a constant whose value depends on the excitation waveform: for square waves, K=1.

Combination of equations (6.4), (6.5) and (6.7) yields

$$L = \frac{N \Delta B A_e}{I} \times 10^8 \tag{6.8a}$$

$$= \frac{0.4 \pi N \Delta B A_e}{H l_e} \times 10^8 \tag{6.8b}$$

Finally, from equation (6.6), we get

$$L = \frac{\mu_o \mu_m N^2 A_e}{l_e} \tag{6.9}$$

where μ_o is the permeability of free space.

If the cross-sectional area per turn of the conductor is A_t and the windings occupy 75% of the total core winding area then, together with equation (6.8a), (6.9) provides the following expression for the required core size

$$A_e A_c = \frac{1.33 \; L \; I_{max} \; A_t}{B_{max}} \times 10^8 \quad . \qquad (6.10)$$

The winding exhibits a DC resistance of magnitude

$$R_{DC} = \frac{\rho_c \; N^2 \; l_w}{A_w \; F_w} \qquad (6.11)$$

where ρ_c is the resistivity of copper, l_w is the mean turn length, A_w is the cross-sectional area of the winding and F_w is the "copper space factor" of the winding = $\frac{N \; \pi \; d^2}{4 \; A_w}$, where d is the copper diameter.

6.3 MAGNETIC SWITCHING

A magnetic switch is a saturable inductor which exhibits a pronounced drop in inductance on saturation of the core material. The technique of magnetic switching utilises the non-linear behaviour of a magnetic material when subjected to a sufficiently strong driving force. By forcing the core material into saturation, the permeability of the material drops to a value of the order of unity. This property can be used to generate short, high peak current pulses which are delayed in time.

The core material of an inductor can exhibit two distinct values of permeability depending on whether the core is saturated or unsaturated. Initially, an inductor operates in the high permeability region of the B-H curve (Figure 6.2); the inductance in this region, given by equation (6.9), is proportional to the unsaturated permeability of the core, μ_{unsat}, which for ferrites is in the range 1000-5000 (Table 6.1). However, once the NI product corresponding to the coercive force at saturation, H_{sat}, is exceeded, the

magnetic field produced tends to align the magnetic domains within the core until they are all parallel. Once core saturation is achieved, the permeability drops to μ_{sat} accompanied by a corresponding drop in inductance. The dominant mode of flux change in saturation is by rotation - the "hard" movement of domains lying in a hard plane of magnetisation due to imperfect grain orientation. No further domain movement occurs until the magnetic field is removed, whereupon the domains move out of alignment and the magnetic induction is increased to the unsaturated level.

The use of saturable inductors in high power pulse generators was first proposed by Melville in 1951[3]. The advent of high power switches, in particular the thyratron, capable of directly switching pulses of short duration and high peak currents resulted in a decline in interest in this method of pulse sharpening. Recently, however, pulsed gas discharge lasers have created a need for reliable switches whose peak and rate of rise of current requirements are at the limit of presently available thyratrons and (asymmetric) thyristors.

Magnetic switching, with the possibility of operation at repetition rates exceeding 10MHz[4], provides a method of prolonging the lifetime of thyratrons in high-frequency, short pulse modulator circuits by one of three methods:

1. Magnetic pulse compression[5-8]
2. Saturable anode inductor[9]
3. Saturable charging inductor.

The operating principles underlying a saturable bypass inductor are described below, followed by a summary of the design parameters relevant to magnetic switching.

Much of the work on magnetic switching for lasers has been directed at magnetic pulse compression (MPC)[8,10] based on saturable inductors. MPC techniques are primarily used for sharpening the front edge of the current pulse at the load. However, as reported here, saturable inductors may also be used to sharpen the fall-time of the current pulse at the load. Although other techniques exist for reducing the fall-time of load current pulses[11], the method reported here provides a means of varying the point in time at which the current pulse terminates.

The magnetic modulator is illustrated in Figure 6.3a. It is based on the capacitive discharge circuit described in Chapter Five, but uses a saturable bypass inductor as the bypass element. The bypass inductor remains unsaturated during the charging cycle. However, application of the discharge pulse causes saturation of the magnetic core at a time determined by the volt-second/ampere-turn product of the inductor (Figure 6.3b). When saturated, the bypass inductor provides an alternative, low impedance path for the discharge current. The rate of fall of laser current is now determined by the relative impedances of the two current paths. The point in time at which saturation occurs can be controlled by biasing the magnetic core. "Positive" biasing, in which the core is pre-biased by means of a secondary source of flux, means that the NI product required to saturate the core is reduced from its full unbiased value. Hence, the core saturates earlier on the bypass element discharge current, and the pulse is terminated earlier. Conversely, saturation of the core, and hence termination of the current pulse, can be delayed by applying a negative flux to the core.

6.4 DESIGN PARAMETERS FOR MAGNETIC SWITCHES

There are several design considerations common to all methods of magnetic switching. These topics are discussed in the order listed below.

1. Saturated and unsaturated inductance
2. Time to saturation
3. Power dissipation
4. Switching jitter
5. Core reset
6. Effect of air gaps.

6.4.1 SATURATED AND UNSATURATED INDUCTANCE

In order to obtain a distinct break between the saturated and unsaturated states, it is desirable to choose a core with as square a B-H curve as possible. A high inductance is required in the initial stage of operation, and a low saturated inductance for rapid transfer of energy. The unsaturated inductance L_{unsat} and the saturated inductance L_{sat} are given by equation (6.9) as

$$L_{unsat} = \frac{\mu_o \mu_{unsat} N^2 A_c}{l_m} \qquad (6.12a)$$

and

$$L_{sat} = \frac{\mu_o \mu_{sat} N^2 A_c}{l_m} \qquad (6.12b)$$

These equations imply that the ratio L_{unsat}/L_{sat} is simply given by

μ_{unsat}/μ_{sat}. In saturation, an ideal toroidal core has an inductance equal to that of an air-cored inductor ie unity. In practice, this value is nearer three. For manganese-zinc ferrites, the initial permeability (below about 1MHz) is typically between 1,000 and 5,000, corresponding to an inductance ratio of the order of 10^3.

6.4.2 TIME TO SATURATION

If it is assumed that leakage through the unsaturated switch is small, the output voltage due to an applied voltage of $V_{sw}(t)$ is zero until saturation. The time to saturation, t_{sat}, is calculated from Faraday's law as the time integral of voltage applied to the switch, or the volt-second rating:

$$V.T = \int_0^{t_{sat}} V_{sw}(t)\, dt = N A_c \Delta B \qquad (6.13)$$

where ΔB is the maximum swing in magnetic induction of the core material. By designing the core to saturate early in the switching cycle, the core cross-sectional area A_c can be reduced.

6.4.3 POWER DISSIPATION

Switch dissipation may be classified as hysteresis, eddy current and copper losses. Hysteresis and eddy current losses occur in the dynamic magnetisation of the material. The former dominates during the delay time and may be calculated from the magnetic volume and the area enclosed by the B-H loop (at the frequency of operation)[12]

$$P_h = \frac{\pi^2 \mu_{sat} g^2}{4 \Delta B^2 k_s} E_t \qquad (6.14)$$

where E_t is the energy transferred and g is the compression ratio of the switch.

Eddy current losses dominate during saturation and are due to I^2R losses in the core laminations, etc. These losses depend on switching time, core material and thickness and switching waveform[12]

$$P_e = \frac{\pi^2 \mu_{sat} \Delta B^3 d^2 g^2}{16 B_{sat} k_s \rho t_{sat}} E_t \qquad (6.15)$$

where ρ and d are the resistivity and thickness of the magnetic material. Eddy current losses are reduced by causing the switch to saturate early in the switching waveform.

6.4.4 SWITCHING JITTER

Jitter in a magnetic system is determined by variations in the time to saturation. There are two effects leading to these variations. From equation (6.13), the time to saturation is dependent on hold-off voltage and magnetic induction swing. Hence, we have

$$\text{Jitter} = \partial t_{sat} = - \frac{\partial V}{V} . t_{sat} + \frac{\partial \Delta B_{sat}}{\Delta B_{sat}} . t_{sat} \quad . \qquad (6.16)$$

The first term is easily controlled by directly regulating the applied voltage. For a 0.1% voltage regulation, and a time to saturation of 1μs, this

would correspond to a time jitter of 1ns. The parameter ΔB_{sat} is a function of temperature and initial magnetisation. Variations are minimised by accurately controlling the temperature (either by forced-air or oil cooling) and by resetting the core to the same point on the B-H curve following saturation. This topic is considered next.

6.4.5 CORE RESET

For reproducible operation and low jitter, the core is forced to reset to the same point on the B-H curve following saturation by re-orientating the magnetic domains. This is achieved by applying a current sufficient to saturate the core in the reverse direction. The magnitude of this current is found from Ampere's circuital law (equation 6.3)

$$i_{reset} = \frac{\int \underline{H}.\underline{dl}}{0.4\ \pi\ N} = \frac{H_{sat}\ l_m}{0.4\ \pi\ N} \quad . \tag{6.17}$$

The time required to reset to this point is expressed in terms of the voltage applied to the core during reset, V_{reset}

$$t_{reset} = \frac{N\ A_c\ B_{sat}}{V_{reset}} \quad . \tag{6.18}$$

6.4.6 EFFECT OF AIR GAPS

The inclusion of an air gap of length l_g in the magnetic path increases the reluctance of the core. Equation (6.3) is modified to

reluctance of the core. Equation (6.3) is modified to

$$\int \underline{H}.\underline{dl} = H_m l_m + H_g l_g = 0.4 \pi NI \qquad (6.19)$$

where the subscripts m and g refer to the conditions within the magnetic material and air gap. Combining equations (6.6) and (6.11) provides an expression for the magnetic induction within the magnetic material

$$H = \frac{0.4 \pi NI \mu_m}{l_m + \mu_m l_g} \qquad (6.20)$$

This implies that for a given NI product, the magnetic induction is reduced by a factor $(l_m/l_m+\mu_m l_g)$.

Hence, the NI product, or magnetising force, required to saturate the core increases due to the presence of an air gap. This has no effect on the time to saturation of the switch but should be avoided due to the increase in unsaturated inductance.

6.5 DESIGN OF MAGNETIC SWITCH AND BIAS CIRCUITRY

6.5.1 MAGNETIC MATERIALS

Two ferrite materials are studied. The first of these is L4A, which is a nickel-zinc ferrite, and the second is lithium-titanium-zinc[14]. The core properties are summarised in Table 6.3. Toroidal cores are preferred due to the relative ease of adding a secondary winding for biasing (Section 6.5.4).

	Toroidal core No.1 (Ni-Zn)	Toroidal core No.2 (Li-Ti-Zn)	Units
Magnetic path length (l_m)	0.25	0.5	m
Cross-sectional area (A_c)	3.5	3.0	cm^2
ΔB_{max} at 25°C	0.8	0.65	T
μ_{unsat}	500-2000	1000-2000	-
μ_{sat}	<5.0	<5.0	-

Table 6.3. Parameters for the two magnetic cores used in the magnetic switch

The unsaturated inductance of the magnetic switch is chosen to have the same value as the conventional bypass inductor of Figure 5.1 (approximately 100μH) since this inductance provides a rate of rise of discharge current pulse that is acceptable for the SVL (approximately 200ns, from Figure 6.1b). In order to decrease the current pulse fall-time, the saturated inductance L_{sat} must be minimised. Equations (6.1a) and (6.1b) show that the ratio L_{unsat}/L_{sat} is simply given by μ_{unsat}/μ_{sat}. For the two core materials considered, this ratio is of the order of 1000.

6.5.2 STATE VARIABLE ANALYSIS OF A MODULATOR CONTAINING A FLUX-CONTROLLED BYPASS INDUCTOR

The modulator circuit of Figure 6.3a is modelled assuming values of circuit components, gas pressure, etc, which correspond to the experimental conditions for the two laser head designs of Chapter Five. The parameters used in the computer model are listed in Table 6.4.

	Value		
	High Heat-loss	Low Heat-loss	Units
Electrode spacing (l_d)	50	80	cm
Active zone length (l_a)	45	75	cm
Optical cavity length (l_c)	150	150	cm
Tube radius (R_d)	13.0	25.4	mm
External helium pressure (p_{He})	200	75	torr
Tube wall temperature (T_w)	620	620	^oC
Storage capacitor charging voltage (V_{cs})	14.4	16.0	kV
Storage capacitance (C_s)	4	4	nF
Unsaturated bypass inductance (L_{unsat})	100	100	μH
Relative permeability of inductor core (μ_{rel})	2000	500	-
Bypass inductor resistance (R_b)	5.0	0.5	Ω

Table 6.4. Parameters for saturable inductor model

The saturation behaviour of the magnetic switch is modelled based on a linear piece-wise approximation to the B-H characteristic of Figure 6.2. The instantaneous inductance of the switch is determined by the magnitude of the flux, which is proportional to the inductor current. In the high permeability region, the unsaturated inductance is set at 100μH. The ratio of unsaturated to saturated permeability is set at 500, yielding a saturated inductance of 200nH. The region between these two conditions is linearly interpolated for values of flux lying between ±20% of the "knee" in Figure 6.2. The model assumes no hysteresis. Losses are included in the resistance of the saturable inductor.

The circuit of Figure 6.3a is defined to the generalised circuit analysis program (GCAP) by means of the file listed in Table 6.5. Element symbols refer to the practical modulator circuit of Figure 3.4 with a saturable inductor L_b.

```
TITLE   SATURABLE BYPASS INDUCTOR
V       Vs
7.2E3   6       1
R       Rs
5       6       7
C       Csm
1E-6    7       1
L       Lc
200E-3  4       7
R       Rdc
1       8       4
R       Rt      VAR
100E3   8       12      0.1
L       Lt
200E-9  12      11
C       Cs
4E-9    15      8
R       Rcs
1       15      19
L       Lcs
200E-12 18      19
R       Rd
0.1     1       25
L       Ld
50E-9   25      24
R       Rb
0.5     18      23
L       Lb      GRA
100E-6  23      24      500     43
R       Rl
SUBA
L       Ll
SUBB
TIME
1E-9    5E-6
PLOT    I       Rl
0       1E-6    20E-9
PLOT    V       Rl
0       1E-6    20E-9
```

Table 6.5. Input file applicable to the electrical circuit of Figure 6.3a

The parameters SUBA and SUBB represent subprograms entered by the user to define the atomic and thermodynamic state variables to GCAP, which in turn determine the resistance and inductance of the plasma. The file listed in Table 6.5 instructs GCAP to perform a transient analysis of the circuit over a period of 5μs with a 1ns increment of calculation.

The laser discharge current and cathode voltage are plotted over a period of 1μs at intervals of 20ns. Figure 6.4 shows the effect of varying the saturation current on the laser discharge current pulse for the high heat-loss (13.0mm diameter) tube. Currents of 20A, 43A, 54A and 60A cause saturation of the bypass inductor on the leading edge, peak, trailing edge and following termination of the discharge current pulse. In each case, the laser current shows a rapid initial fall to a level determined by the relative impedances of the discharge and bypass element. The resistance of the laser plasma at this point is indicated in Figure 6.4. This is followed by a period of very much slower current fall.

A more satisfactory situation is shown in Figure 6.5 for the 25.4mm diameter tube. The helium pressure has been chosen to produce the discharge current pulse shown. The break between the two regions is less pronounced. The discharge current increases until the inductor saturates and draws away the bulk of the current. The peak current within the laser head is 250A, whilst the rise-time and fall-time are 115ns (10%-90%) and 70ns (90%-10%), respectively. The rate of fall of current is 2.9×10^{9} As^{-1}, which is below the specification in Table 1.2. There is a current reversal which continues until the inductor current reduces to a level at which the inductor emerges from saturation and, due to the unidirectional nature of the CX1826 thyratron, the circuit current circulates within the laser head and bypass inductance. The magnitude of the

peak inverse current is 50A, or 20% of the forward current. The decay of this current is characterised by a time constant determined by the inductance and resistance of the two components. The former is dominated by the unsaturated bypass inductance. The resistance is predominantly that of the laser plasma which has a value of approximately 30Ω at peak inverse current (360ns after thyratron switching) and 11Ω one microsecond into the afterglow. During this time, the L/R time constant is of the order of 5μsec.

The electron temperature follows the laser cathode voltage during the development of electric field, E, across the laser head, reaching a peak of 4.3eV, until the voltage across the discharge tube swings negative (Figure 6.6). Since, the predominant source term for electron temperature during the discharge current pulse is proportional to E^2 (Equation 3.58), T_e follows the electric field down to a minimum of 1.3eV, and then rises to equilibriate at approximately 3.0eV. The electron temperature decays with the same time constant as the discharge current and is maintained above 1eV for several microseconds into the afterglow period. Consequently, as shown in Figure 6.7, at no point in the afterglow period is a population inversion achieved.

6.5.3 CONSTRUCTION OF MAGNETIC SWITCH

Two saturable bypass inductors are constructed using the available ferrite cores, each with an unsaturated inductance of approximately 100μH (based on the performance of the air-cored inductor). The first saturable inductor consists of two cores of type No. 1 (Table 6.3) sandwiched together, and wound with ten turns of high-voltage cable (25kV). The second saturable inductor consists of two parallel windings, each of ten turns, wound on a single L4A toroid of type No. 2 in Table 6.3. Since, from equation (6.2), the time to saturation

increases linearly with the number of turns, the use of parallel windings reduces the effective number of turns and hence the time to saturation. The unsaturated and saturated inductances of this switch are 90μH and 180nH based on a relative permeability of 500.

6.6 EXPERIMENTAL RESULTS

Two methods of biasing the magnetic core are tested and these are described below.

6.6.1 PULSE BIASING

The circuit used to pulse bias the core of type No. 2, shown in Figure 6.8, operates as follows. Trigger units No. 1 and No. 2 are EEV sub-modulators designed primarily for triggering large thyratrons and capable of supplying a 2kV voltage pulse. Trigger unit No. 2, in addition to triggering the thyratron, also sends a 15 volt pre-pulse to trigger unit No. 1. After a variable time-delay, a current pulse is applied to the bias winding of the saturable inductor. With a value of 100 ohms, a current pulse of 20A was produced. For the core materials available, saturation occurs with an applied magnetic force of several hundred ampère-turns per metre. Hence, for l_m=0.5m, and three turns as the bias winding, a 20A current pulse will have a significant effect upon core saturation.

This technique enabled variation of the time to core saturation over the entire duration of the laser current pulse. Figure 6.9 shows the discharge and bypass inductor current pulse in 16mbar of helium. The magnitude of the forward current pulse is 480A and the fall-time is 80ns, implying a rate of fall of

current of 6.0×10^9 As^{-1}. These figures satisfy the modulator requirements of Table 1.2 for strontium laser operation. However, the peak inverse voltage is 85% of the forward voltage. Furthermore, as the delay is reduced such that the bias pulse coincides with the laser discharge current pulse, interaction between the circuit components caused distortion of the current pulse applied to the bias winding. Pulse biasing was therefore abandoned in favour of DC biasing which is described below.

6.6.2 DC BIASING

DC biasing provides a more reliable and succesful method of controlling the time to saturation of the ferrite core. The circuit shown in Figure 6.10 can deliver up to 100A into a three turn bias winding. This is sufficient to saturate the core. The L_1/C_1 filter prevents high-voltage spikes induced in the bias winding from damaging the low-voltage circuitry. The laser tube is of 25.4mm diameter operated at 70 mbar.

Results obtained for zero bias current are presented in Figure 6.11. This shows that the ferrite saturates with zero DC bias current at a time shortly after the peak of the laser current pulse. This reduces the laser current fall-time from 275ns (with the air-cored bypass inductor) to 96ns. The peak current is 250A.

The application of DC bias current allows control over the point in time at which saturation occurs. A DC bias current of 1-2A was sufficient to reduce further the laser current fall-time to 67ns, as indicated in 6.12a and 6.12b. The magnitude and rate of fall of the peak forward current are in precise agreement with the model predictions described in Section 6.4.2.

6.6.3 A METHOD OF INHIBITING CURRENT REVERSAL

Figure 6.12a reveals that using a parallel saturating bypass inductor increases the current reversal in the laser which is present in the air-cored bypass inductor modulator circuit. This reversal has a magnitude of approximately 50A, which is about 20% of the forward current pulse, in good agreement with the model predictions of Section 6.4.2. The electric field does not go to zero immediately following the discharge current pulse. The presence of the inverse current through the laser provides a source of heating for the free electrons in the discharge which may adversely affect the laser output as previously discussed.

A solution has been successfully applied to inhibit current reversal in a conventional SVL circuit[15], in which a solid-state diode is placed in parallel with the laser head, as indicated schematically in Figure 6.13. The diode is a high-current (500mA), fast-recovery (50ns) clipping diode (Type No. ED6877[16]) with a 60kV hold-off voltage. The diode provides an alternative low-impedance path for inverse current.

6.6.4 CONCLUSIONS

The use of a saturable bypass inductor to terminate the current pulse in a strontium vapour laser discharge tube has been demonstrated. The magnetic modulator is capable of generating current pulses of amplitude 250A and current densities of $325Acm^{-2}$. The fall-time using an air-cored inductor was 275ns. This was reduced to 96ns with zero DC core bias current, and further reduced to 67ns with 1-2A of DC bias current. A 20% current reversal may be removed by

placing a high-current, fast-recovery solid-state diode in parallel with the laser tube. Precise choice of laser operating parameters could then be used to produce a rapid fall-time discharge current pulse with minimal overshoot.

Adjustment of the DC bias current provides control over the point at which the core saturates. For a fixed set of laser operating parameters, DC bias may not be required, since the saturable bypass inductor could be designed to saturate at, or near, the peak of the laser current pulse.

REFERENCES FOR CHAPTER 6

1. Mullard Limited, Book 3, 'Components, Materials and Assemblies', Part 3, "Vinkor inductor cores", Mullard House, Torrington Place, London, England. WC1E 7HD (1984).

2. Marconi Electronic Services Limited, 'MEDL Microwave Ferrite Materials', Microwave Division, Doddington Road, Lincoln, England. LN6 3LF.

3. W. S. Melville, "The use of saturable reactors as discharge devices for pulse generators", Proc. IEE (London), 98, Part 3 (Radio and Communication), No.53, pp.185-207 (1951).

4. D. L. Birx, L. L. Reginato and J. A. Schmidt, "An investigation into the repetition rate limitations of magnetic switches", IEEE Fifteenth Power Modulator Symposium, pp.4-8, Buffalo, NY, June, 1982.

5. I. Smilanski, "Electrical excitation of an XeCl laser using magnetic pulse compression", Appl. Phys. Lett., 40 (7), pp.547-548 (1982).

6. T. J. Pacala, I. S. McDermid and J. B. Laudenslager, "Ultra-narrow linewidth, magnetically switched, long pulse, xenon chloride laser", Appl. Phys. Lett., 44, pp.658-660 (1984).

7. R. A. Petr, J. F. Zumdieck, J. Demboski, I. Smilanski, J. J. Ewing and R. E. Center, "Magnetic pulse compression for copper vapour lasers", IEEE Fourth Pulsed Power Conference, pp.236-241, Albuquerque, NM (1983).

8. I. Smilanski, "Advances in magnetic pulse compression for copper vapour lasers", IEEE Eighteenth Power Modulator Symposium, pp.84-85, Hilton Head, SC, June 20-22, 1988.

9. C. J. Eichenauer, "A magnetic assist to hydrogen thyratron switch tubes", Fourth Hydrogen Thyratron Symposium, p.3, Fort Monmouth, NJ, Nov 17-18, 1955.

10. S. E. Ball, "Optimum switching time for magnetic switches", IEEE Eighteenth Power Modulator Symposium, pp.86-89, Hilton Head, SC, June 20-22, 1988.

11. J. P. O'Loughlin and J. Sidler, "The end-of-line tail biter", Sixth IEEE Pulsed Power Conference, pp.692-695, Arlington, VA, June 29-July 1, 1987.

12. H. Tanaka and M. Obara, "Efficiency characterization of repetition-rated magnetic modulators for pumping TEA CO_2 lasers and KrF lasers", Sixth IEEE Pulsed Power Conference, pp.719-722, Arlington, VA, June 29-July 1, 1987.

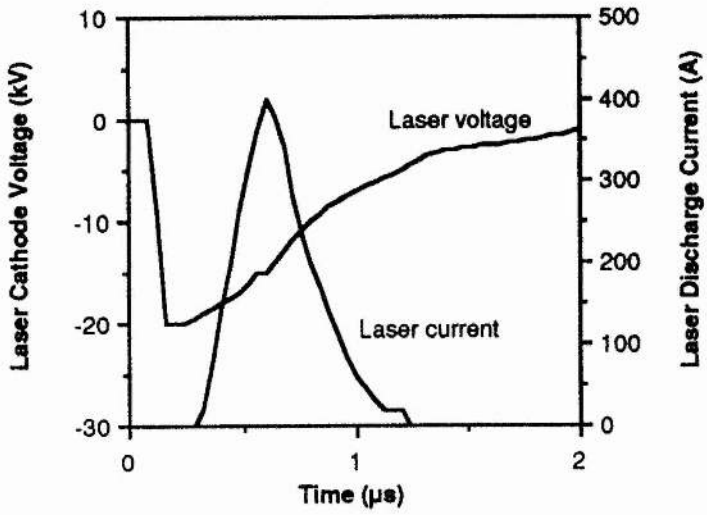
13. H. J. Baker, P. A. Ellsmore and E. C. Sille, "An efficient laser pulser using ferrite magnetic switches", J. Phys. E: Sci. Instrum., 21, pp.218-224 (1988).

14. Marconi Electronic Services Limited, 'MEDL Microwave Ferrite Materials', Microwave Division, Doddington Road, Lincoln, England. LN6 3LF.

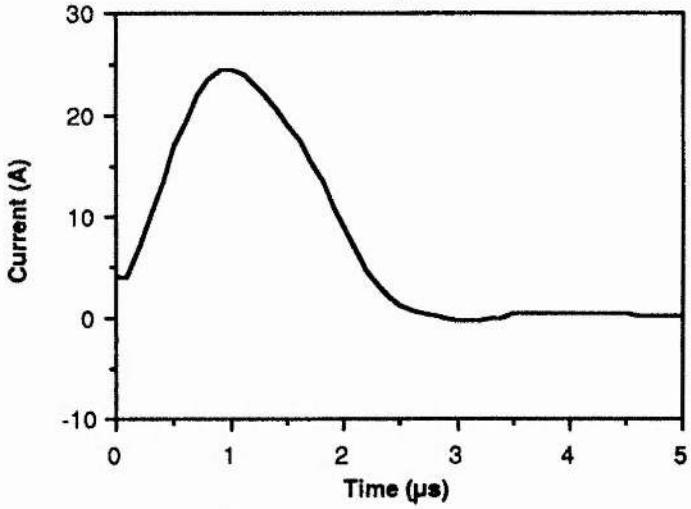
15. C. E. Little and J. A. Piper, "High-power violet Sr^+ recombination lasers", SPIE vol.1041, *Metal Vapour, Deep Blue, and Ultraviolet Lasers*, pp.167-174, Los Angeles, CA (1989).

16. Electronic Devices, Inc., 21, Gray Oaks Avenue, Yonkers, NY 10710.

FIGURES FOR CHAPTER 6



(a) Laser cathode voltage
and discharge current



(b) Bypass inductor current

Circuit parameters :	Capacitor charging voltage	= 16 kV
	Storage capacitance	= 4 nF
	Bypass inductance	= 100 μH

Figure 6.1. Voltage and current waveforms obtained with an air-cored inductor.

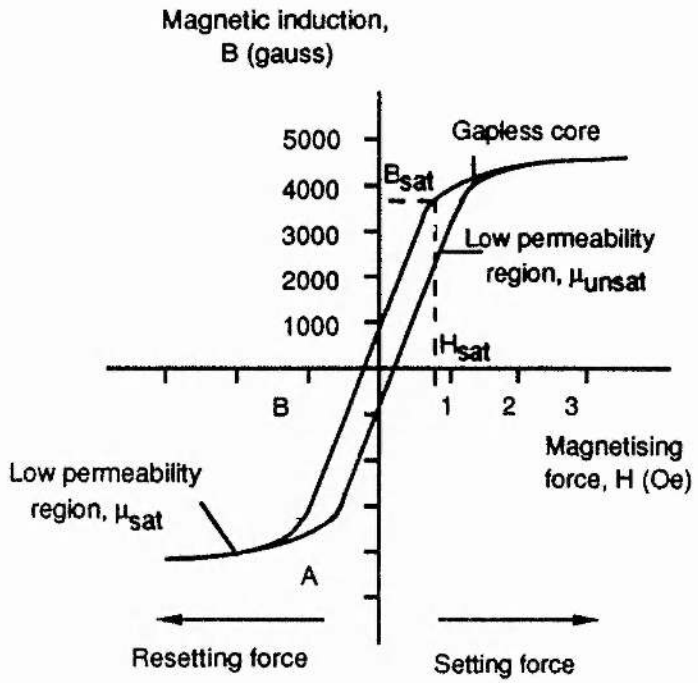
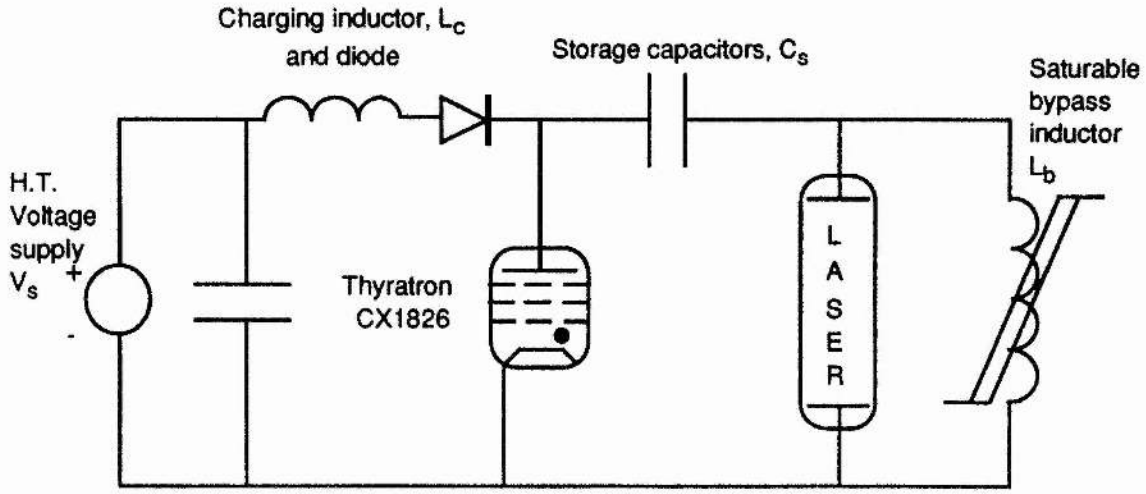
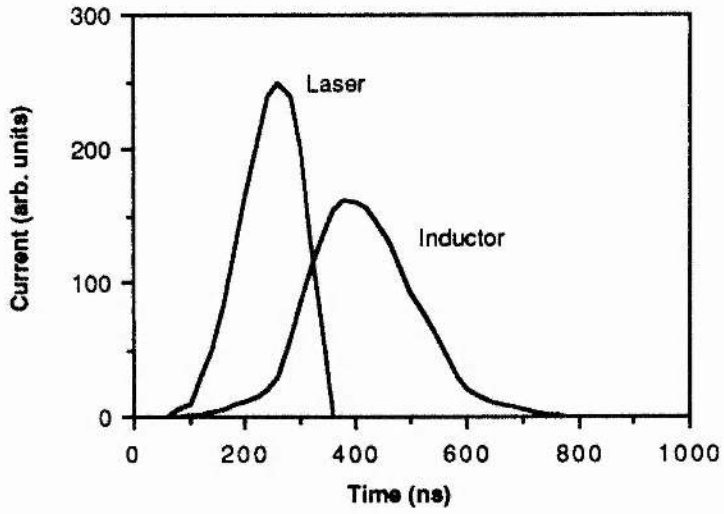


Figure 6.2. Magnetisation of low-loss power ferrite material 3C8 (after Ref.1)



(a)



(b)

Figure 6.3. (a) The saturable inductor modulator schematic
(b) Bypass inductor saturation

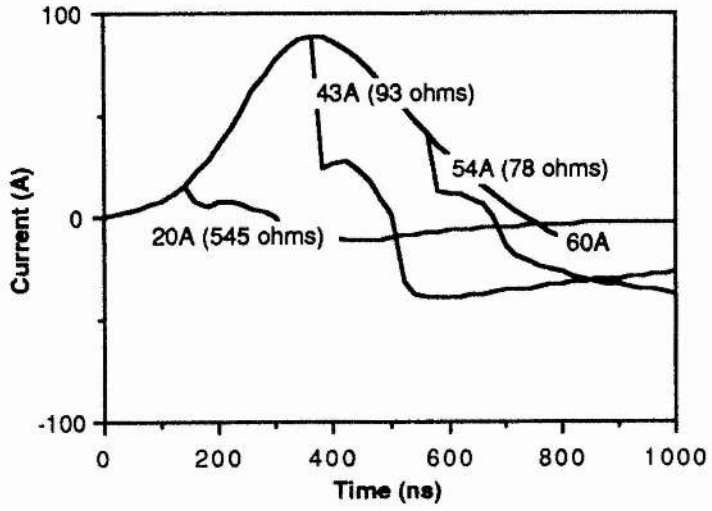


Figure 6.4. Predicted discharge current for a 13.0mm id discharge tube for various values of inductor saturation flux

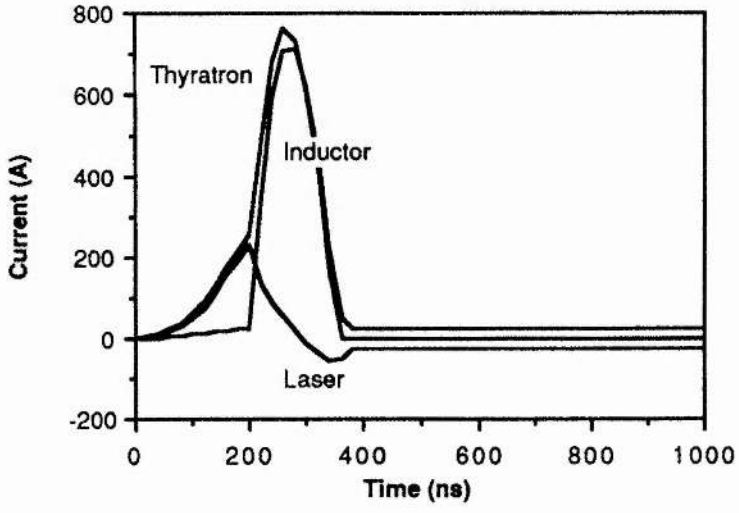


Figure 6.5. Current waveforms for a 25.4mm id discharge tube

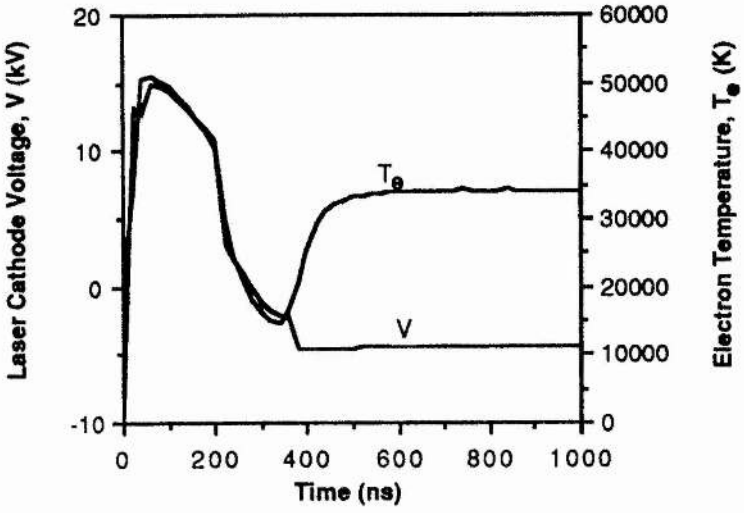


Figure 6.6. Laser cathode voltage and electron temperature for a 25.4mm id discharge tube

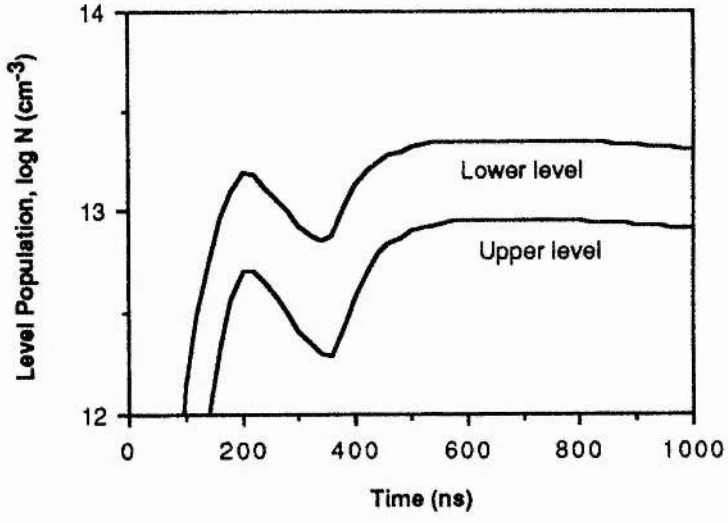


Figure 6.7. Upper and lower laser level populations for a 25.4mm id discharge tube

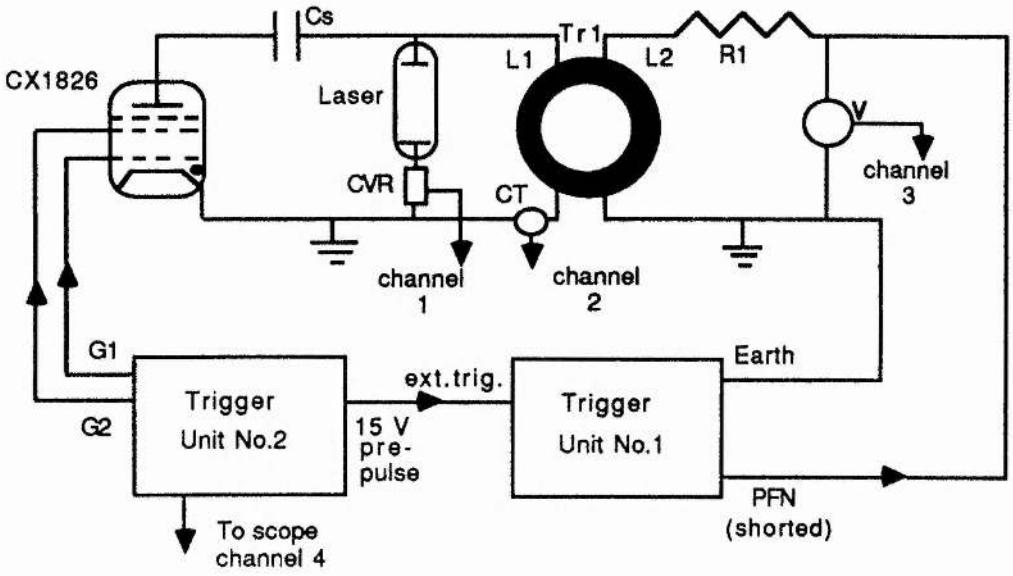


Figure 6.8. Pulse biased magnetic pulse compression with saturable bypass inductor (secondary windings of toroidal core)

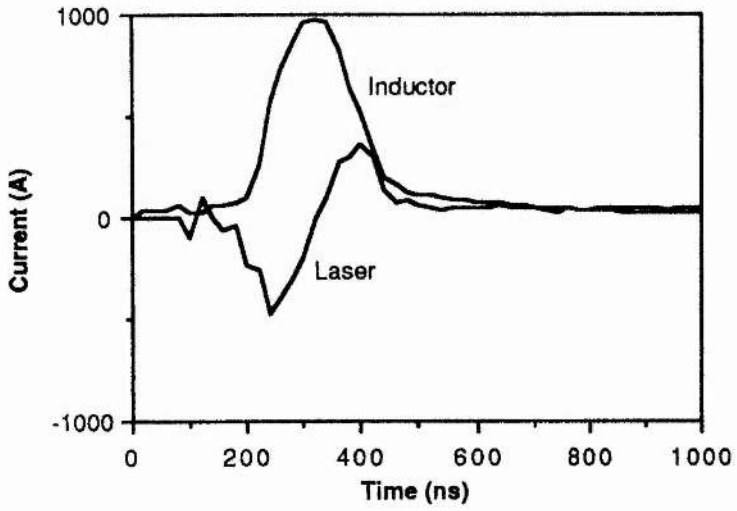


Figure 6.9. Laser and bypass inductor current waveforms for inductor based on core No.2

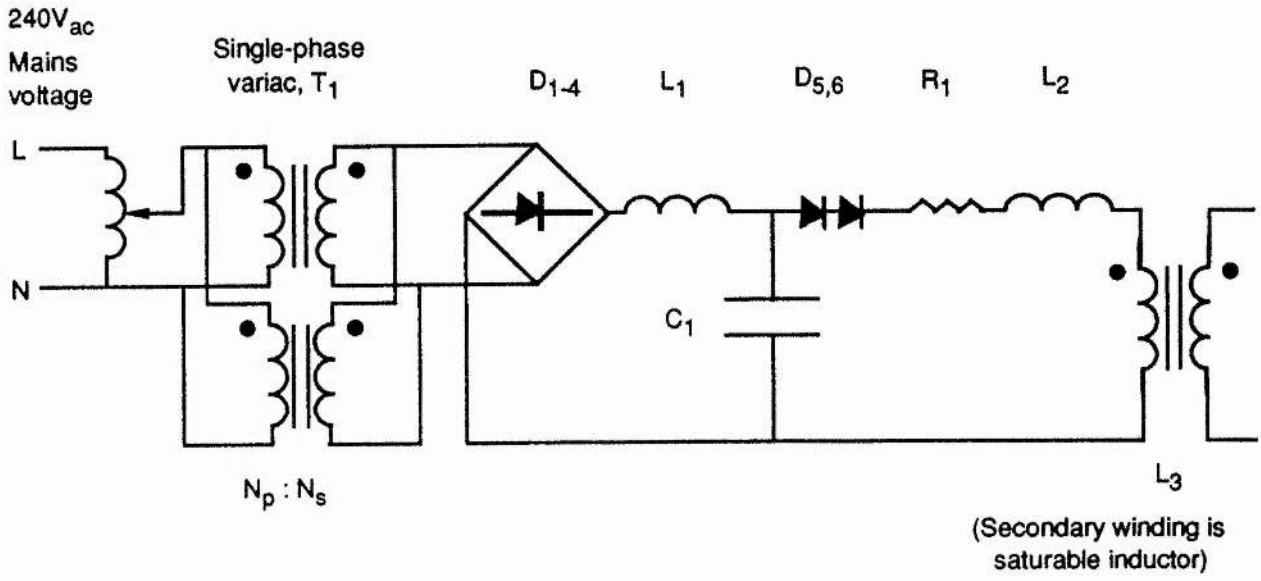
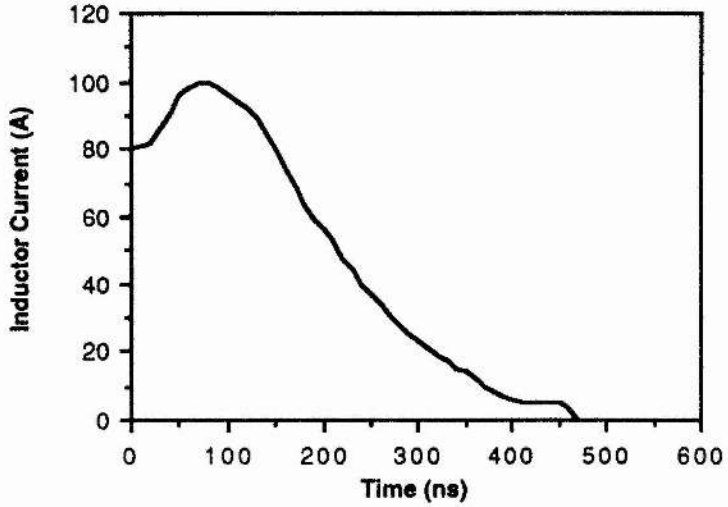
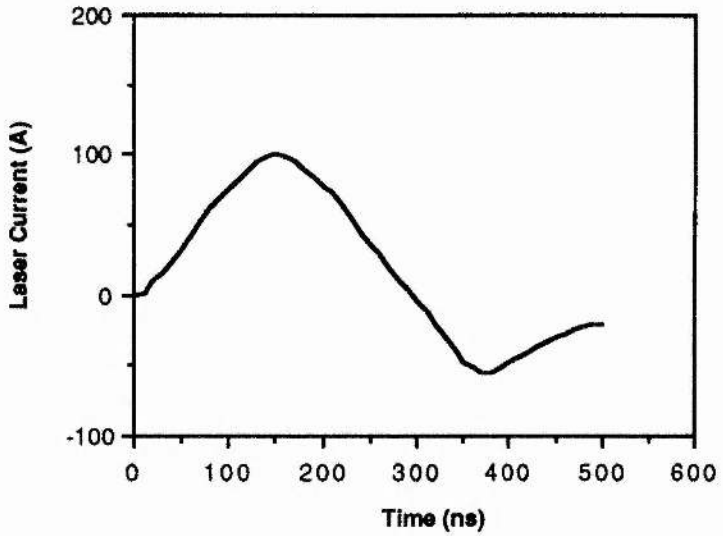


Figure 6.10. DC biasing of saturable inductor

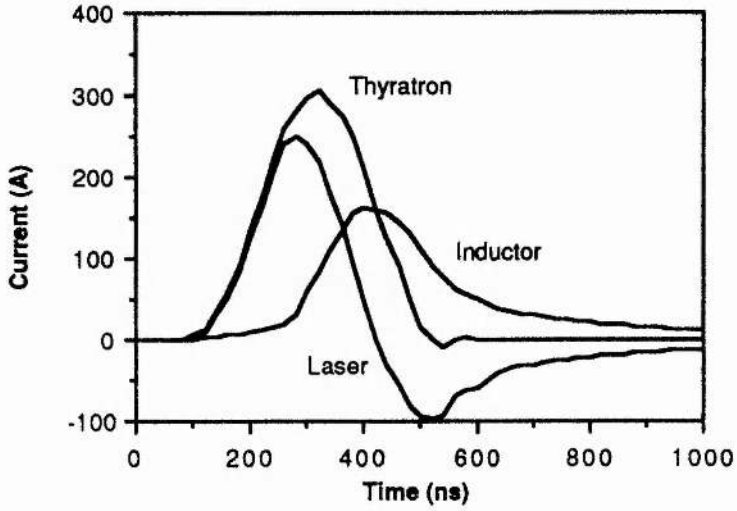


(a) Saturable inductor current

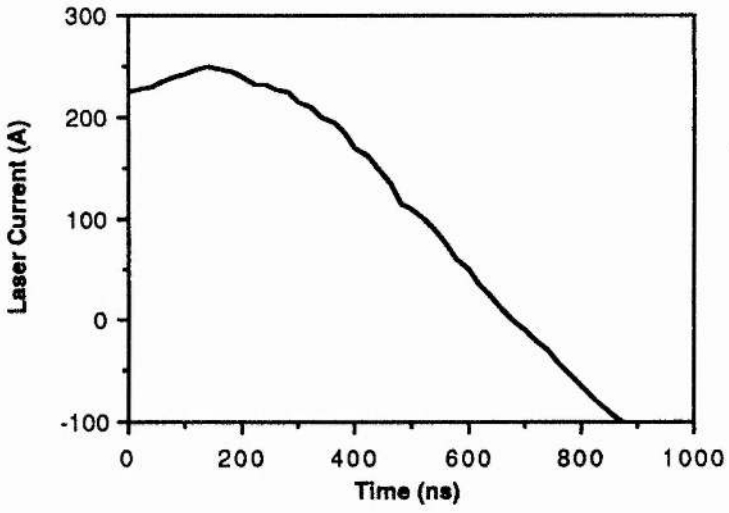


(b) Laser current

Figure 6.11. Current waveforms for inductor based on core No.1; zero bias current



(a) Saturable inductor, thyratron and laser current



(b) Detail of laser current

Figure 6.12. Current waveforms for inductor based on core No.1; bias current = $1.1A_{DC}$

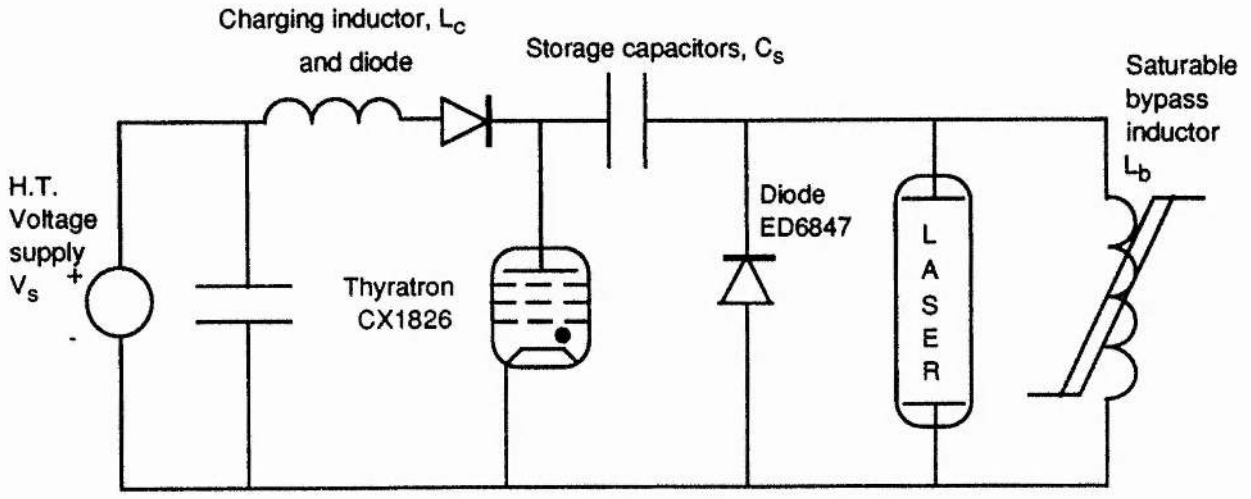


Figure 6.13. The saturable inductor modulator including diode to inhibit current reversal

A UNIFIED STATE VARIABLE ANALYSIS OF REPETITIVELY PULSED RECOMBINATION LASERS

Appendices to a thesis presented by

Andrew Keith Kidd, B. Sc., M. Sc.,

to the

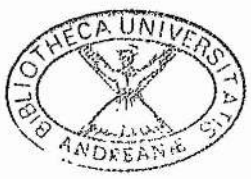
University of St. Andrews

in application for the degree of

Doctor of Philosophy

A. K. Kidd

12th November, 1991



Th B113

A UNIFIED STATE VARIABLE ANALYSIS OF REPETITIVELY PULSED RECOMBINATION LASERS

Contents of Appendices

Page

APPENDIX E Generalised Circuit Analysis Program Program (GCAP) Listing (Continued)

Appendix A

Numerical Solution Algorithms Employed by GCAP

1. NEWTON-RAPHSON ALGORITHM FOR SOLUTION OF NON-LINEAR DIFFERENTIAL STATE EQUATIONS

Solutions of the system of non-linear state equations (2.3a) at different instants in time are independent of each other. Hence, suppressing the time variable t reduces the problem to that of solving the following system of n non-linear state equations

$$\begin{aligned} \dot{x}_1(t) &= f_1(x_1, x_2, ..., x_n, u_1, u_2, ..., u_n) \\ &\vdots \qquad\quad \vdots \qquad\quad \vdots \qquad\quad \vdots \\ \dot{x}_n(t) &= f_n(x_1, x_2, ..., x_n, u_1, u_2, ..., u_n) \end{aligned}$$

or,

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}, \mathbf{u}) \qquad \text{NON-LINEAR STATE EQUATION} \qquad (A.1)$$

A properly-modelled non-linear resistive circuit will have either a unique solution or a finite number of solutions. The most general approach to obtaining these solutions is by numerical iteration. Assume that the function $\mathbf{f}(\mathbf{x}, \mathbf{u})$ has a Taylor series expansion about any point $(\mathbf{x}^{(j)}, \mathbf{u}^{(j)})$ where

$$\mathbf{x}^{(j)} \sim \left[x_1^{(j)}, x_2^{(j)}, ..., x_n^{(j)}\right]^T \qquad (A.2)$$

such that

$$f(x) = f(x^{(j)}) + J(x^{(j)})(x - x_i^{(j)}) + \text{higher order terms} \qquad (A.3)$$

where the n x n square matrix of first partial derivatives $J(x^{(j)})$ is the Jacobian matrix.

The Newton-Raphson formula[1]

$$x^{(j+1)} = x^{(j)} - \left[J(x^{(j)})\right]^{-1} f(x^{(j)}) \qquad (A.4)$$

forms the basis of the Newton-Raphson algorithm for solution of equations (A.1), described below.

Choose a small positive number ε such that the algorithm is terminated after (j+1) iterations whenever we have

$$\| x^{(j+1)} - x^{(j)} \| \sim \left\{ \sum_{p}^{i=1} \left[x_i^{(j+1)} - x_i^{(j)} \right]^2 \right\}^{0.5} < \varepsilon \quad . \qquad (A.5)$$

Then, $x = x^{(j+1)}$ is taken to be a solution of equation (A.1).

Step 0. Choose a maximum number of steps m.
Choose a tolerance $\varepsilon > 0$.
Make an initial estimate $x^{(0)}$.
Set j = 0.

Step 1. Solve the linear equation (A.4) for $x^{(j+1)}$.

Step 2. If the inequality (A.5) holds, then terminate; else set j = j+1.
If j+1 > m, then terminate; else repeat steps 1 and 2.

The algorithm requires a sufficiently accurate initial estimate $x^{(0)}$ for

convergence. GCAP assumes a figure of one-half of the element value in the absence of excitation. If convergence is not achieved within ten iterations, then the algorithm is repeated with a value of 50% of that of the original estimate. If convergence is still not achieved, and there is less tendency to converge, then a value of 50% is added to the previous estimate and the algorithm repeated. This process is repeated until convergence is achieved.

2. RUNGE-KUTTA METHOD OF SOLUTION OF DIFFERENTIAL STATE EQUATIONS

The Runge-Kutta method is based on the Taylor series expansion of a function. This refers to the most commonly-employed fourth-order Runge-Kutta algorithm:

$$x_o = x(t_o) \tag{A.6}$$

and

$$x_{k+1} = x_k + h\phi(x_k,t_k;h) \tag{A.7}$$

where

$$\phi(x_k,t_k;h) = \frac{1}{6}\left[\alpha_k + 2\beta_k + 2\chi_k + \delta_k\right] \tag{A.8}$$

and h is the step size.

Application to the differential state equations (2.1) yields the four coefficients

$$\alpha_k = Ax_k + Bu_k \tag{A.9a}$$

$$\beta_k = A\left[x_k + \frac{h}{2}\left\{Ax_k + Bu_k\right\}\right] + Bu_{k+1/2}$$

$$= \left[A + \frac{h}{2}A^2\right]x_k + \frac{h}{2}ABu_k + Bu_{k+1/2} \tag{A.9b}$$

$$\chi_k = A\left[x_k + \frac{h}{2}\left\{\left[A + \frac{h}{2}A^2\right]x_k + \frac{h}{2}ABu_k + Bu_{k+1/2}\right\}\right] + Bu_{k+1/2}$$

$$= \left[A + \frac{h}{2}A^2 + \left[\frac{h}{2}\right]^2 A^3\right]x_k + \left[\frac{h}{2}\right]^2 A^2 Bu_k + \left[\frac{h}{2}AB + B\right]u_{k+1/2} \qquad \text{(A.9c)}$$

$$\delta_k = A\left[x_k + h\left\{\left[A + \frac{h}{2}A^2 + \left[\frac{h}{2}\right]^2 A^3\right]x_k + \left[\frac{h}{2}\right]^2 A^2 Bu_k + \frac{h}{2}AB + B\right]u_{k+1}\right\}\right]$$
$$+ Bu_{k+1}$$

$$= \left[A + \frac{h}{2}A^2 + \left[\frac{h}{2}\right]^2 A^3 + \left[\frac{h}{2}\right]^3 A^4\right]x_k + \left[\frac{h}{2}\right]^3 A^3 Bu_k$$
$$+ \left[\left[\frac{h}{2}\right]^2 A^2 B + hAB\right]u_{k+1/2} + Bu_{k+1} \qquad \text{(A.9d)}$$

where we have

$$t_{k+1/2} = t_o(k+1/2) \qquad \text{(A.10)}$$

and

$$u_{k+1/2} = u(t_{k+1/2}) \qquad \text{(A.11)}$$

Hence, by combining equations (A.7), (A.8) and (A.9), we get

$$x_{k+1} = x_k + \frac{h}{6}\left[\alpha_k + 2\beta_k + 2\chi_k + \delta_k\right]$$

$$= \left[1 + hA + \frac{1}{2!}h^2A^2 + \frac{1}{3!}h^3A^3 + \frac{1}{4!}h^4A^4\right]x_k$$

$$+ \frac{1}{6}\left[1 + hA + \frac{1}{2}h^2A^2 + \frac{1}{4}h^3A^3\right]Bu_k$$

$$+ \frac{2}{3}\left[1 + \frac{h}{2}A + \frac{1}{8}h^2A^2\right]Bu_{k+1} \qquad \text{(A.12)}$$

REFERENCE FOR APPENDIX A

1. L. O. Chua, C. A. Desoer and E. S. Kuh, 'Linear and Nonlinear Circuits', McGraw-Hill (1987).

Appendix B

Reaction Rates and Cross-sections of Processes Included in the Strontium Vapour Laser Model

Process	Rate (k_{ij})/ cross-section (σ_{ij})	Reference	Description
A. Strontium			
(a) Electron-impact processes			
1. $Sr^+ + e^- \rightarrow Sr_1^+ + e^-$		a	Electron-impact excitation
2. $Sr^+ + e^- \rightarrow Sr_1,^+ + e^-$		a	" " "
3. $Sr_m^+ + e^- \rightarrow Sr_1^+ + e^-$		a	" " "
4. $Sr_m,^+ + e^- \rightarrow Sr_1^+ + e^-$		a	" " "
5. $Sr_m,^+ + e^- \rightarrow Sr_1,^+ + e^-$		a	" " "
6. $Sr_1^+ + e^- \rightarrow Sr_2^+ + e^-$		a	" " "
7. $Sr_1,^+ + e^- \rightarrow Sr_2^+ + e^-$		a	" " "
157. $Sr_1^+ + e^- \rightarrow Sr^{+*} + e^-$		a	" " "
8. $Sr_2^+ + e^- \rightarrow Sr^{+*} + e^-$		a	" " "
9. $Sr + e^- \rightarrow Sr^* + e^-$		a	" " "
10. $Sr_1^+ + e^- \rightarrow Sr^+ + e^-$	Inverse of 1	b	Superelastic collision
11. $Sr_1,^+ + e^- \rightarrow Sr^+ + e^-$	Inverse of 2	b	" "
12. $Sr_1^+ + e^- \rightarrow Sr_m^+ + e^-$	Inverse of 3	b	" "
13. $Sr_1^+ + e^- \rightarrow Sr_m,^+ + e^-$	Inverse of 4	b	" "
14. $Sr_1,^+ + e^- \rightarrow Sr_m,^+ + e^-$	Inverse of 5	b	" "
15. $Sr_2^+ + e^- \rightarrow Sr_1^+ + e^-$	Inverse of 6	b	" "
16. $Sr_2^+ + e^- \rightarrow Sr_1,^+ + e^-$	Inverse of 7	b	" "
17. $Sr^{+*} + e^- \rightarrow Sr_2^+ + e^-$	Inverse of 8	b	" "
18. $Sr^* + e^- \rightarrow Sr + e^-$	Inverse of 9	b	" "
19. $Sr + e^- \rightarrow Sr^+ + e^- + e^-$		a	Electron-impact ionisation
20. $Sr^* + e^- \rightarrow Sr^+ + e^- + e^-$		a	" " "
151. $Sr + e^- \rightarrow Sr^{++} + e^- + e^- + e^-$	$7.5 \times 10^{-16} cm^2$	a	" " "
21. $Sr^+ + e^- \rightarrow Sr^{++} + e^- + e^-$		a	" " "
22. $Sr^{+*} + e^- \rightarrow Sr^{++} + e^- + e^-$		a	" " "

23.	$Sr^{++}+e^-+e^- \rightarrow Sr^{+*}+e^-$	$8 \times 1.8 \times 10^{-8}\ LT_e^{-9/2}$	1	Collisional-radiative recombination
24.	$Sr^{+}+e^-+e^- \rightarrow Sr^{*}+e^-$	$1.8 \times 10^{-8}\ LT_e^{-9/2}$	1	" "
25.	$Sr^{++}+e^- \rightarrow Sr^{+*}+h\upsilon_{3+*}$		c	Radiative recombination
26.	$Sr^{+}+e^- \rightarrow Sr^{*}+h\upsilon_{1*}$		c	" "

(b) Optical processes

27.	$Sr_2^{+}+h\upsilon_{21} \rightarrow Sr_1^{+}+2h\upsilon_{21}$		d	Stimulated emission (λ=430.5nm)
28.	$Sr_2^{+}+h\upsilon_{21}, \rightarrow Sr_1^{+},+2h\upsilon_{21},$		d	" " (λ=416.2nm)
29.	$Sr_2^{+} \rightarrow Sr_1^{+}+h\upsilon_{21}$	$1.30 \times 10^{8}\ sec^{-1}$	e	Spontaneous emission (λ=430.5nm)
30.	$Sr_2^{+} \rightarrow Sr_1,^{+}+h\upsilon_{21},$	$7.1 \times 10^{7}\ sec^{-1}$	e	" " (λ=416.2nm)
31.	$Sr_1^{+} \rightarrow Sr^{+}+h\upsilon_{10}$	$1.46 \times 10^{8}\ sec^{-1}$	e	" " (λ=407.8nm)
32.	$Sr_1,^{+} \rightarrow Sr^{+}+h\upsilon_{1'0}$	$1.20 \times 10^{8}\ sec^{-1}$	e	" " (λ=421.6nm)
33.	$Sr_1^{+} \rightarrow Sr_m^{+}+h\upsilon_{1m}$	$6.9 \times 10^{6}\ sec^{-1}$	e	" " (λ=1032.7nm)
34.	Reabsorption of $h\upsilon_{10}$		2,f	Radiation trapping

B. **Helium**

(a) Electron-impact processes

35.	$He+e^- \rightarrow He_m+e^-$	$6.2 \pm 2.0 \times 10^{-18}\ cm^2$	3,a	Electron-impact excitation
36.	$He+e^- \rightarrow He^{*}+e^-$	$4.5 \times 10^{-18}\ cm^2$	a,g	" " "
37.	$He_m+e^- \rightarrow He^{*}+e^-$	$4.5 \times 10^{-18}\ cm^2$	a,g	" " "
38.	$He_m+e^- \rightarrow He+e^-$	Inverse of 35	b	Superelastic collision
39.	$He^{*}+e^- \rightarrow He+e^-$	Inverse of 36	b	" "
40.	$He^{*}+e^- \rightarrow He_m+e^-$	Inverse of 37	b	" "
41.	$He_2^{*}+e^- \rightarrow He+He+e^-$	$3.0 \times 10^{-7}\ cm^3 sec^{-1}$	4,h	Electron quenching
42.	$He+e^- \rightarrow He^{+}+e^-+e^-$	$0.6 \times 10^{-16}\ cm^2$	5	Electron-impact ionisation
43.	$He_m+e^- \rightarrow He^{+}+e^-+e^-$	$6.5 \times 10^{-16}\ cm^2$	6	" " "
44.	$He^{*}+e^- \rightarrow He^{+}+e^-+e^-$		g	" " "
45.	$He^{+}+e^- \rightarrow He^{++}+e^-+e^-$	$0.05 \times 10^{-16}\ cm^2$	4	" " "
46.	$He^{+}+e^-+e^- \rightarrow He+e^-$	Not of interest (too slow)	i	Collisional-radiative recombination
47.	$He^{+}+e^-+e^- \rightarrow He_m+e^-$	" " "	i	" "

48.	$He^+ + e^- + e^- \rightarrow He^* + e^-$	$7.1 \times 10^{-20} (300/T_e)^{9/2}$	7	Collisional-radiative recombination
49.	$He_2^+ + e^- + e^- \rightarrow He^* + He + e^-$	6×10^{-20} cm^{6}sec^{-1}	g	" "
50.	$He_2^+ + e^- + e^- \rightarrow He_2^* + e^-$	$1.8 \times 10^{-8} (300/T_e)^{9/2}$	1	" "
51.	$He^+ + e^- \rightarrow He + h\upsilon$	Not of interest (too slow)	8,i	Radiative recombination
52.	$He^+ + e^- \rightarrow He_m + h\upsilon$	" " "	9,i	" "
53.	$He^+ + e^- \rightarrow He^* + h\upsilon$	$4.2 \times 10^{-12} (300/T_e)^{0.7}$	10	" "
54.	$He_2^+ + e^- \rightarrow (He_2^*)_* \rightarrow He^* + He \rightarrow He + He + h\upsilon$	$8.9 \pm 0.5 \times 10^{-9} (T_e/300)^{1/2}$	11	Two-body dissociative recombination

(b) Atom-atom collisional processes

55.	$He + He_m \rightarrow He_m + He$	No change to population	i	Atom-metastable collisions
56.	$He_m + He + He \rightarrow He_2^* + He$	0.35 cm^3.torr^2.sec^{-1}	12	Three-body collisional de-excitation
57.	$He + He_m \rightarrow He + He$	$6 \times 10^{-15} (T_{He}/300)^{1/2}$	13	Atom-metastable deactivation
58.	$He_m + He_m \rightarrow He + He^+ + e^-$	$7.1 \times 10^{-8} T_{He}^{1/6}$	12	Metastable-metastable collisional ionisation
59.	$He^+ + He \rightarrow He + He^+$	No change to population	i	Charge exchange (Penning ionisation)
60.	$He^+ + He_m \rightarrow He_m + He^+$	" " " "	i	" "
61.	$He^+ + He \rightarrow He^+ + He^+ + e^-$	16×10^{-17} cm^2	14,j	
62.	$He^+ + He \rightarrow He^{++} + He^+ + e^- + e^-$	33×10^{-17} cm^2	14,j	
63.	$He^+ + He + e^- \rightarrow He_2^+ + e^-$		15	
64.	$He^+ + He + He \rightarrow He_2^+ + He$	1.0×10^{-31} cm^{6}sec^{-1}	16	Molecular ion-neutral association
65.	$He_m + He_m \rightarrow He_2^+ + e^-$	Unimportant @ > 5 torr	17	
66.	$He_m + He_m + He \rightarrow He_2^+ + He + e^-$	10^{-10} cm^{3}sec^{-1}	18	
67.	$He^+ + He + e^- \rightarrow He^* + He$		19	Neutral-assisted collisional-radiative recombination
68.	$He_2^+ + He + e^- \rightarrow He_2^* + He$	$4.0 \pm 0.5 \times 10^{-20} (T_e/293)^{1/2}$ cm^{6}sec^{-1}	16	" "

(c) Optical processes

69.	$He^* \rightarrow He+h\upsilon$		20,e	Spontaneous emission
70.	$He^* \rightarrow He_m+h\upsilon$		20,e	" "
71.	$He_2^* \rightarrow He_2^{*'} \rightarrow He_2^{**} + h\upsilon$		20,e	" "

(d) Transport processes

72.	He → walls		k	Diffusion to walls, then wall deactivation
73.	He_m → walls	(520/p) $cm^2.torr.sec^{-1}$	21,l	" " " "
74.	He^* → walls	μ_o=17.3±0.7$cm^2V^{-1}sec^{-1}$	15,l	" " " "
75.	He^+ → walls	(560±20/p) $cm^2torr.sec^{-1}$	22,l	" " " "
76.	He_2^+ → walls	(724/p) $cm^2.torr.sec^{-1}$	23,l	" " " "

C. Neon

(a) Electron-impact processes

77.	$Ne+e^- \rightarrow Ne_m+e^-$	9.1 ± 2.0 x 10^{-18} cm^2	24,a	Electron-impact excitation
78.	$Ne+e^- \rightarrow Ne^*+e^-$	Inverse of 81	a	" " "
79.	$Ne_m+e^- \rightarrow Ne^*+e^-$	Inverse of 82	25,a	" " "
80.	$Ne_m+e^- \rightarrow Ne+e^-$	Inverse of 77	b	Superelastic collision
81.	$Ne^*+e^- \rightarrow Ne+e^-$	1.0 x 10^{-13} cm^2	25,b	" "
82.	$Ne^*+e^- \rightarrow Ne_m+e^-$	1.0 x 10^{-13} cm^2	25,b	" "
83.	$Ne_2^*+e^- \rightarrow Ne+Ne+e^-$	3.0 x 10^{-7} cm^3sec^{-1}	26	Electron quenching
84.	$Ne+e^- \rightarrow Ne^++e^-+e^-$	1.8 x 10^{-16} cm^2	5	Electron-impact ionisation
85.	$Ne_m+e^- \rightarrow Ne^++e^-+e^-$	6.0 x 10^{-16} cm^2	5	" " "
86.	$Ne^*+e^- \rightarrow Ne^++e^-+e^-$		g	" " "
87.	$Ne^++e^- \rightarrow Ne^{++}+e^-+e^-$	0.42 x 10^{-16} cm^2	5	" " "
88.	$Ne^++e^-+e^- \rightarrow Ne+e^-$	Not of interest (too slow)	i	Collisional-radiative recombination
89.	$Ne^++e^-+e^- \rightarrow Ne_m+e^-$	" " "	i	" "
90.	$Ne^++e^-+e^- \rightarrow Ne^*+e^-$	7.1 x 10^{-20} $(300/T_e)^{9/2}$	7	" "
91.	$Ne^++e^- \rightarrow Ne^*+h\upsilon$		27	Radiative recombination
92.	$Ne^++e^- \rightarrow Ne_m+h\upsilon$	Not of interest (too slow)	i	" "

93.	$Ne^+ + e^- \rightarrow Ne + h\upsilon$	Not of interest (too slow)	i	Radiative recombination
94.	$Ne_2^+ + e^- \rightarrow (Ne_2^*) + h\upsilon \rightarrow$	2.2×10^{-7} cm^{3}sec^{-1}	28	Two-body dissociative
	$Ne^* + Ne \rightarrow Ne + Ne + h\upsilon$	$3.7 \times 10^{-8} T_e^{-0.43}$ cm^{3}sec^{-1}	29	recombination

(b) Atom-atom collisional processes

95.	$Ne + Ne_m \rightarrow Ne_m + Ne$	No change to population	i	Atom-metastable collisions
96.	$Ne_m + Ne + Ne \rightarrow Ne_2^* + Ne$	4.1×10^{-34} cm^{6}sec^{-1}	30	Three-body collisional de-excitation
97.	$Ne^* + Ne \rightarrow Ne_m + Ne$	7.0×10^{-11} cm^{3}sec^{-1}	4	Neutral-atom collisional de-excitation
98.	$Ne_m + Ne_m \rightarrow Ne + Ne^+ + e^- \rightarrow Ne_2^+ + e^-$	1.4×10^{-18} cm^2	24	Metastable-metastable collisional ionisation
99.	$Ne^+ + Ne \rightarrow Ne + Ne^+$	No change to population	i	Charge exchange (Penning ionisation)
100.	$Ne^+ + Ne_m \rightarrow Ne_m + Ne^+$	" " " "	i	" "
101.	$Ne^+ + Ne + Ne \rightarrow Ne_2^+ + Ne$	$4.4 \pm 0.4 \times 10^{-32}$ cm^{6}sec^{-1}	31, 32	Molecular ion-neutral association

(c) Optical processes

102.	$Ne^* \rightarrow Ne_m + h\upsilon$	5.24×10^7 sec^{-1} (2p-1s)	33,1	Spontaneous emission
		1.30×10^7 sec^{-1} (3s-2p)	34,1	" "
103.	$Ne_2^* \rightarrow Ne + Ne + h\upsilon$	7.5×10^7 sec^{-1}	30,1	" "
104.	$Ne^* + h\upsilon \rightarrow Ne^+ + e^-$	2.3×10^{-18} cm^2	35	Absorption
105	$Ne_2^+ + h\upsilon \rightarrow Ne^+ + Ne$	1.9×10^{-17} cm^2	36	"

(d) Transport processes

106.	$Ne_m \rightarrow$ walls	$170 \pm 10 (T_e/300)^{0.73}$ cm^2.torr.sec^{-1}	25, 37,1	Diffusion to walls, then wall deactivation
107.	$Ne^+ \rightarrow$ walls	$\mu_o = 4.1$ cm^{2}V^{-1}sec^{-1}	38,1	" " " "
108.	$Ne_2^+ \rightarrow$ walls	$\mu_o = 6.5$ cm^{2}V^{-1}sec^{-1}	38,1	" " " "

D. **Helium-Strontium**

(a) Atom-atom collisional processes

109. $He^+ + Sr \rightarrow He + Sr^{++} + e^-$	2.0×10^{-15} cm^2	39,m	Atom-atom charge exchange
110. $He^+ + Sr \rightarrow He + Sr^+$		39,i	Atom-atom charge exchange
111. $He^+ + Sr^+ \rightarrow He + Sr^{++}$		39,i	" " " "
112. $He_2^+ + Sr \rightarrow He + He + Sr^{++} + e^-$		39,m	Molecule-atom charge exchange
113. $He_m + Sr \rightarrow He + Sr^{++} + e^- + e^-$	2.0×10^{-15} cm^2	39,i	Penning ionisation
114. $He_m + Sr^+ \rightarrow He + Sr^{++} + e^-$		39,i	" "

E. **Neon-Strontium**

(a) Atom-atom collisional processes

115. $Ne^+ + Sr \rightarrow Ne + Sr^{++} + e^-$	0.5×10^{-15} cm^2	40,m	Atom-atom charge exchange
116. $Ne^+ + Sr \rightarrow Ne + Sr^+$		40,i	" " " "
117. $Ne^+ + Sr^+ \rightarrow Ne + Sr^{++}$		40,i	" " " "
118. $Ne_2^+ + Sr \rightarrow Ne + Ne + Sr^{++} + e^-$		40,m	Molecule-atom charge exchange
119. $Ne_m + Sr \rightarrow Ne + Sr^{++} + e^- + e^-$	0.5×10^{-15} cm^2	40,i	Penning ionisation
120. $Ne_m + Sr^+ \rightarrow Ne + Sr^{++} + e^-$		40,i	" "

F. **Helium-Neon**

(a) Atom-atom collisional processes

121. $He_m + Ne \rightarrow He + Ne^*$	6.46×10^{-11} cm^{3}sec^{-1}	41	Resonant transfer excitation exchange
122. $He + Ne_m \rightarrow He + Ne^*$	1.9×10^{-14} cm^{3}sec^{-1}	25	" "
123. $He + Ne^* \rightarrow He + Ne_m$	1.5×10^{-18} cm^2	g	Atom-atom collisional de-excitation
124. $He_m + Ne \rightarrow He + Ne$		42	" " "
125. $He + Ne^* \rightarrow He_m + Ne$	2.53×10^{-10} cm^{3}sec^{-1}	41	Resonant transfer excitation exchange
126. $He^+ + Ne \rightarrow He + Ne^+$	7.1×10^{-16} cm^2	43	Slow ion production
127. $He^+ + Ne \rightarrow He^+ + Ne^+ + e^-$	6.0×10^{-16} cm^2	g	

128. $He_2^+ + Ne \rightarrow He + He + Ne^+$	1.5×10^{-15} cm^2	15	Molecule-atom charge exchange
129. $He + Ne^+ + Ne \rightarrow He + Ne_2^+$		27,j	Associative ionisation
$\rightarrow (HeNe)^+ + Ne$	$3\pm1 \times 10^{-31}$ cm^{6}sec^{-1}	44	
130. $He_m + Ne \rightarrow (HeNe)^+ + e^-$		45	Associative ionisation
131. $He + He + Ne^+ \rightarrow$			
$(HeNe)^+ + He$	2.1×10^{-32} cm^{6}sec^{-1}	27	" "
132. $(HeNe)^+ + e^- \rightarrow$		45,	Dissociative recombination
$(HeNe)^* \rightarrow He^* + Ne^*$		46	
133. $(HeNe)^+ + Ne \rightarrow He + Ne_2^+$	3.0×10^{-11} cm^{3}sec^{-1}	27,m	

(b) Transport processes

134. $Ne_m \rightarrow$ walls (in He)	310 ± 20 cm^2.torr.sec^{-1}	37,l	Diffusion to walls, then wall deactivation
135. $Ne^+ \rightarrow$ walls (in He)	322 ± 32 cm^2.torr.sec^{-1}	44,m	" " " "
136. $(HeNe)^+ \rightarrow$ walls (in He)	810 ± 40 cm^2.torr.sec^{-1}	27,n	" " " "

Notes

a. Reaction rate calculated from equation (3.28)
b. " " " " " (3.29)
c. Probability of radiative recombination is very much lower than that of collisional-radiative recombination
d. Reaction rate of stimulated emission calculated from equation (3.51)
e. Spontaneous transition probability obtained by method of Bates and Damgaard (Section 3.4.1(A))
f. Emission coefficient given by equation (3.52)
g. Estimated
h. By analogy to process No.83 for neon
i. Negligible
j. Reaction rate calculated from equation (3.28) with v_e replaced by v_{atom}
k. Only important if neon present
l. Transport processes discussed in Section 3.7
m. Reaction rate calculated by method of Smirnov and Firsov
n. Reaction rate of process No.133 can be deduced from transport measurements.

Table B.1. Reaction rates and cross-sections

REFERENCES FOR APPENDIX B

1. I. S. Veselovskii, "Electron recombination coefficient with three-body collisions in a plasma", Sov. Phys. - Tech. Phys., 14, No.2, p.193, August, 1969.

2. T. Holstein, "Imprisonment of resonance radiation in gases", Phys. Rev., 72, No.12, p.1212 (1947).

3. W. L. Borst, "Excitation of metastable argon and helium atoms by electron impact", Phys. Rev., A9, No.3, p.1195 (1974).

4. L. A. Levin, S. E. Moody, E. L. Klosterman, R. E. Center and J. J. Ewing, "Kinetic model for long pulse XeCl laser performance", IEEE J. Quantum Electron., QE-17 (12), p.2282 (1981).

5. L. Vriens, "Calculation of the absolute ionisation cross-sections of He, He^*, He^+, Ne, Ne^*, Ne^+, Ar, Ar^*, Hg and Hg^*", Phys. Lett., 8, No.4, p.260 (1964).

6. D. R. Long and R. Geballe, "Electron-impact ionization of He ($2s^3S$)", Phys. Rev., A1, No.2, p.260 (1970).

7. D. R. Bates and A. Dalgarno, "Electronic Recombination", in 'Atomic and Molecular Processes', ed. D. R. Bates, Academic Press, New York, pp.245-257 (1962).

8. J. Berlande, M. Cheret, R. Deloche, A. Gonfalone and C. Manus, "Pressure and electron density dependence of the electron-ion recombination coefficient in helium", Phys. Rev., A1, No.3, p.887 (1970).

9. D. R. Bates, "Electron recombination in helium", Phys. Rev., 77, p.718 (1950).

10. M. A. Biondi, "Recombination", in 'Principles of Laser Plasmas', ed. G. Bekefi, Wiley-Interscience, New York, pp.125-158 (1976).

11. C. L. Chen, C. C. Leiby and L. Goldstein, "Electron temperature dependence of the recombination coefficient in pure helium", Phys. Rev., 121, No.5, p.1391 (1961).

12. I. Ya. Fugol', O. N. Grigoraschenko and D. A. Myshkis, "Experimental investigation of the destruction of metastable atoms of helium in a plasma at low temperatures", Sov. Phys. - JETP, 33, No.1, p.227 (1971).

13. A. C. G. Mitchell and M. W. Zemansky, 'Resonance Radiation and Excited Atoms', Cambridge University, New York (1971).

14. R. A. Langley, D. W. Martin, D. S. Harmer, J. W. Hooper and E. W. McDaniel, "Cross sections for ion and electron production in gases by fast helium ions (0.133-1.0MeV). I. Experimental", Phys. Rev., 136, No.2A, p.A379 (1964).

15. H. J. Oskam, "Microwave investigation of disintegrating gaseous discharge plasmas", Phillips Res. Rept., 13, pp.335-457 (1958).

16. R. Deloche, P. Monchicourt, M. Cheret and F. Lambert, "High-pressure helium afterglow at room temperature", Phys. Rev., A13, No.3, p.1140 (1976).

17. C. S. Willett, "Pulsed discharges", in 'Introduction to Gas Lasers: Population Inversion Mechanisms', Pergamon Press, section 3.5, pp.91-101 (1974).

18. P. L. Pakhomov, G. P. Reznikov and I. Ya. Fugol, "The afterglow of helium in the plasma of a high-frequency pulse discharge at 77°K", Optics Spectrosc., 20, p.5 (1966).

19. L. P. Pitaevskii, "Electron recombination in a monatomic gas", Sov. Phys. - JETP, 15, No.5, p.919 (1962).

20. S. Bashkin and J. O. Stoner, 'Atomic Energy Levels and Grotrian Diagrams. Vol.I. Hydrogen I - Phosphorus XV', North Holland, Amsterdam (1975).

21. M. A. Biondi, "Ionization by the collision of pairs of metastable atoms", Phys. Rev., 82, p.453 (1951).

22. A. V. Phelps and S. C. Brown, "Positive ions in the afterglow of a low pressure helium discharge", Phys. Rev., 86, No.1, p.102 (1952).

23. R. W. Huggins and J. H. Cahn, "Metastable measurements in flowing helium afterglow", J. Appl. Phys., 38 (1), p.180 (1967).

24. S. N. Salinger and J. E. Rowe, "Determination of the predominant ionization and loss mechanisms for the low voltage arc mode in a neon plasma diode", J. Appl. Phys., 39, p.4299 (1968).

25. A. V. Phelps, "Diffusion, de-excitation and three-body collision coefficients for excited neon atoms", Phys. Rev., 114, No.4, p.1011 (1959).

26. J. H. Jacob, J. C. Hsia, J. A. Morgan and M. Rokni, "Pulse shape and laser-energy extraction from electron-beam pumped KrF", J. Appl. Phys., 50 (8), pp.5130-5134 (1979).

27. G. E. Veatch and H. J. Oskam, "Recombination and ion-conversion processes in helium-neon mixtures", Phys. Rev., A2, No.4, p.1422 (1970).

28. H. J. Oskam and V. R. Mittelstadt, "Recombination coefficient of molecular rare-gas ions", Phys. Rev., 132, No.4, p.1445 (1963).

29. C. H. Brau, "Rare gas halide lasers", in 'Excimer Lasers', ed. C. K. Rhodes, Springer-Verlag, Berlin, p.105 (1979).

30. D. L. Huestis, R. M. Hill, H. H. Nakano and D. C. Lorents, "Quenching of Ne^*, F^* and F_2^* in $Ne/Xe/NF_3$ and $Ne/Xe/F_2$ mixtures", J. Chem. Phys., 69, pp.5133-5139 (1978).

31. A. P. Vitols and H. J. Oskam, "Reaction rate constant for $Ne^+ + 2Ne \rightarrow Ne_2^+ + Ne$", Phys. Rev., A5, No.6, p.2618 (1972).

32. V. S. Egorov and A. M. Shukhtin, "The afterglow and its relation to gas density in a pulse discharge in Ne", Optics Spectrosc., 9, p.419 (1960).

33. W. R. Bennett and P. J. Kindlmann, "Radiative and collision-induced relaxation of atomic states in the $2p^53p$ configuration of neon", Phys. Rev., 149, No.1, pp.38-51 (1966).

34. R. Arrathoon, "Positive column population calculations for the evaluation of dispersive effects in He-Ne lasers", J. Appl. Phys., 40 (7), p.2875 (1969).

35. C. Duzy and H. A. Hyman, "Photoionization of excited rare gas atoms", Phys. Rev., A22, pp.1878-1883 (1980).

36. H. H. Michaels and R. H. Hobbs, "Electronic structure of the noble gas dimer ions. II. Theoretical absorption spectrum for the $A^2\Sigma^+_{1/2u} \rightarrow D^2\Sigma^+_{1/2g}$ system", J. Chem. Phys., 71, pp.5053-5062 (1979).

37. J. R. Dixon and F. A. Grant, "Decay of the triplet P levels of neon", Phys. Rev., 107, No.1, p.118 (1957).

38. H. J. Oskam and V. R. Mittelstadt, "Ion mobilities in helium, neon and argon", Phys. Rev., 132, No.4, p.1435 (1963).

39. B. M. Smirnov and O. B. Firsov, "Ionization of an atom colliding with an excited atom", Sov. Phys. - JETP Lett., 2, p.297 (1965).

40. B. M. Smirnov, "Excitation transfer in atomic collisions", Sov. Phys. - JETP, 24 (2), p.314 (1967).

41. A. L. Schmeltekopf and F. C. Fehsenfeld, "De-excitation rate constants for helium metastable atoms with several atoms and molecules", J. Chem. Phys., 53 (8), p.3173 (1970).

42. R. T. Young, C. S. Willett and R. T. Maupin, "Effect of helium on population inversion in the helium-neon laser", J. Appl. Phys., 41 (7), p.2936 (1970).

43. C. F. Barnett and P. M. Stier, "Charge exchange cross sections for helium ions in gases", Phys. Rev., 109, No.2, p.385 (1958).

44. P. Lukac, "Afterglow study in helium-neon mixture", J. Phys. D., 3, p.1689 (1970).

45. V. S. Egorov, Y. G. Kozlov and A. M. Shukhtin, "The concentration of excited atoms in a helium-neon pulsed discharge", Optics Spectrosc., 15, p.458 (1963).

46. E. I. Shtyrkov and E. V. Subbes, "Characteristics of pulsed laser action in helium-neon and helium-argon mixtures", Optics Spectrosc., 21, p.143 (1966).

Appendix C

Conference Papers

The following papers were presented at international conferences.

1. A. K. Kidd, C. A. Pirrie, P. D. Culling and C. R. Weatherup, "A method of rapidly terminating the current pulses applied to recombination lasers", in proceedings of SPIE vol. 1046, *Pulse Power for Lasers II*, pp.35-42, Los Angeles, California, Jan 15-20, 1989.

2. A. K. Kidd and A. Maitland, "G C A P : A generalised circuit analysis program for pulsed power", in proceedings of Seventh IEEE Pulsed Power Conference, pp.471-474, Monterey, California, June 11-14, 1989.

3. A. K. Kidd and A. Maitland, "Computer modelling of metal vapour laser modulator circuits", in proceedings of ICPIG XIX (International Conference on Phenomena in Ionised Gases), vol. 2, pp.486-487, Belgrade, Yugoslavia, July 10-14, 1989.

4. A. K. Kidd and A. Maitland, "The application of state variables to the simultaneous analysis of laser circuits and gas discharge populations", presented as additional information at ICPIG XIX (International Conference on Phenomena in Ionised Gases), Belgrade, Yugoslavia, July 10-14, 1989.

1046 05.

A method of rapidly terminating the current pulses applied to recombination lasers

Andrew K. Kidd
Department Of Physics and Astronomy, University of St. Andrews,
St. Andrews, Fife, Scotland. KY16 8DQ

Colin A. Pirrie, Paul D. Culling and Clifford R. Weatherup
English Electric Valve Co. Ltd., 106, Waterhouse Lane,
Chelmsford, Essex, England. CM1 2QU

ABSTRACT

This paper describes the design, construction and test results of a thyratron modulator circuit for use with high peak power metal vapour lasers in which the discharge current pulse through the laser is rapidly terminated. The circuit generates current pulses with peak amplitudes up to 300A and fall times of less than 70ns (90%-10%). The system consists of a thyratron-switched capacitive discharge circuit in which the bypass element is a saturable inductor. Saturation of the magnetic core provides an alternative path for the discharge current. Furthermore, the point in time at which saturation occurs can be controlled by appropriate magnetic core biasing. This paper describes the operation of the circuit.

1. INTRODUCTION

There is interest in pulsed metal vapour lasers for a wide variety of applications, due primarily to their high peak output powers and pulse repetition frequencies. To date, the bulk of the research effort has centred on gold and copper, the former being capable of providing up to 10W in the red (λ = 628nm) and the latter up to a total of 100W in the green (λ = 510.6nm) and the yellow (λ = 578nm). However, despite their relatively modest average power (less than 2W), strontium and calcium lasers represent valuable sources of high peak power radiation respectively in the blue (λ = 430.5nm) and near u.v. (λ = 373.7nm) regions of the spectrum. The two major causes of the low average output power can be explained with reference to the metal vapour laser modulator circuit of Figure 1. As in the case of copper and gold vapour lasers, a population inversion is achieved by discharging a capacitor bank through the laser head. In this case, a current pulse is applied to a high pressure (200 mbar - 1 atm.) helium-neon buffer gas mixture containing small pieces of pure strontium metal. This heats the laser cavity to a temperature sufficient to produce a suitable vapour pressure of the metal for lasing. However, the temperature range for stimulated emission is very narrow and in order to produce a thermally homogeneous plasma, the radial temperature gradients must be reduced. This places a limitation on bore size and hence the active volume for a given length of discharge tube. Thermal relaxation processes can be accelerated by employing rectangular cross-section discharge tubes due to the proximity of the walls. This is presently being investigated[1].

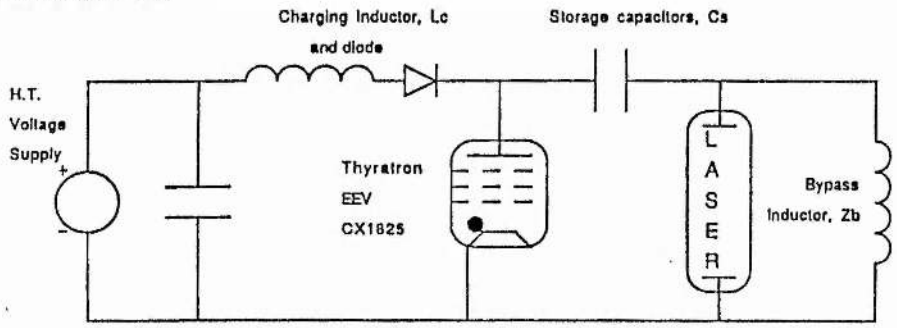


Figure 1. The Laser Modulator Circuit

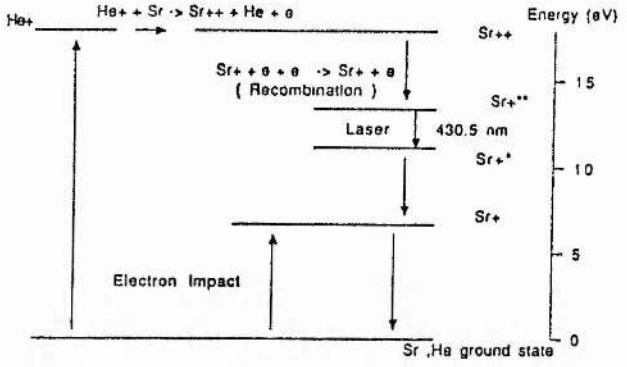


Figure 2. Energy Level Diagram for the Strontium Vapour Laser

The second limitation requires a brief explanation of the mechanism by which the population inversion is produced. A large population of Sr^{++} ions are created during the current pulse by electron-impact. Strontium and calcium lasers are both examples of recombination lasers in which the population inversion is not achieved during the discharge current pulse risetime, but rather by the three-body recombination (e-e-Sr++) of multiply-ionised species as they cascade through the excited states to the ground state in the discharge afterglow (Figure 2). The rate of this recombination process is highly temperature dependent ($\alpha Te^{-9/2}$, where Te is the electron temperature). Rapid termination of the current pulse applied to the laser head enhances cooling of the free electron temperature and allows recombination to proceed. A slowly-decaying current pulse will only serve to re-populate the lower-lying states, including the lower laser level, and destroy the population inversion. It is therefore important to produce a high amplitude current pulse (to achieve a large Sr^{++} density) with as rapid a termination as possible. A clearer indication of the requirements of the modulator circuit is obtained by kinetic coding of the strontium laser. Computer analysis[2,3] has revealed that for a narrow-bore, high-pressure strontium vapour laser, the optimum voltage and current applied to the laser head should be 20-30kV and approximately 1kA, with a risetime of 100-200ns. The analysis revealed that stimulated emission would be enhanced if the fall time were reduced. This is supported by a general review paper[4] on amplification in recombination lasers which concludes that a population inversion may be obtained if the free electrons are cooled rapidly.

The modulator requirements are summarized in Table 1.

Anode Voltage	: 20-30kV
Peak Forward Current	: 250 -> 1400A
Overshoot Current	: Minimal
Pulse Repetition Frequency	: 100Hz - 10kHz
Rate of Rise of Current	: 10^9 A/S (10%-90%)
Rate of Fall of Current	: 5 x 10^9 A/S (90%-10%)

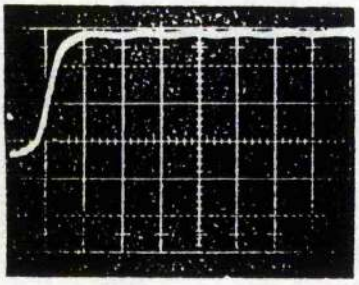
Table 1. Strontium Laser Modulator Requirements

2. ELECTRICAL CIRCUIT

Circuit design was based on the capacitive discharge circuit shown in Figure 1. This is similar to the C-to-C transfer circuit successfully applied to copper vapour lasers [5]; however, the peaking capacitor is removed as this can distort the current pulse falling edge. The circuit consists of a high-voltage power supply which resonantly charges the storage capacitance C_s via a charging inductor L_c and the bypass element Z_b, which is

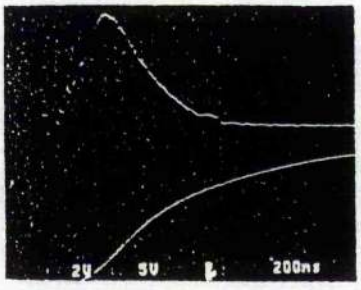
either a resistor or an inductor. The value of Z_b is normally chosen to minimise current flow through Z_b during the discharge pulse. The thyratron used was an EEV CX1825. If the value of Z_b is high enough, an insignificant amount of the discharge current will flow through it. The voltage across the laser increases until the gas breaks down, at which point C_s discharges through the laser.

Figure 3 shows four waveforms obtained operating the circuit with an air-cored inductor as the bypass element.



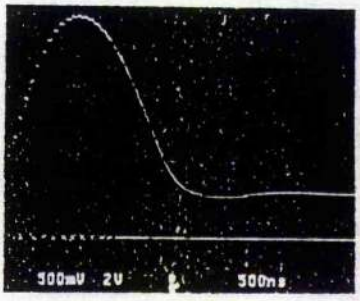
(i) Thyratron charging voltage.

Vertical scale : 5 kV/div
Horizontal scale : 0.01ms/div



(ii) Laser discharge current (Upper trace) / Laser cathode voltage

Vert : 100A,5kV/div
Horiz : 200ns/div



(iii) Bypass inductor current

Vert : 50A/div
Horiz : 500ns/div

Circuit parameters : Capacitor charging voltage = 16 kV
Storage capacitance = 4 nF
Bypass inductor = 100 μH

Figure 3. Voltage and Current Waveforms obtained with an Air-cored Inductor.

The laser current fall-time of Figure 3(ii) is approximately 275ns (90%-10%), which corresponds to a rate of fall of current considerably slower than the optimum value indicated in Table 1.

Although other techniques exist for reducing the fall time of load current pulses[6], the method reported here uses a saturable inductor as the bypass element, as indicated in Figure 4a. The bypass inductor remains unsaturated during the charging cycle. However, application of the discharge pulse causes saturation at a time determined by the volt-second/ampere-turn product of the inductor. When saturated, the bypass inductor provides an alternative low impedance path for the discharge current. The rate of fall of laser current is now determined by the relative impedances of the two current paths.

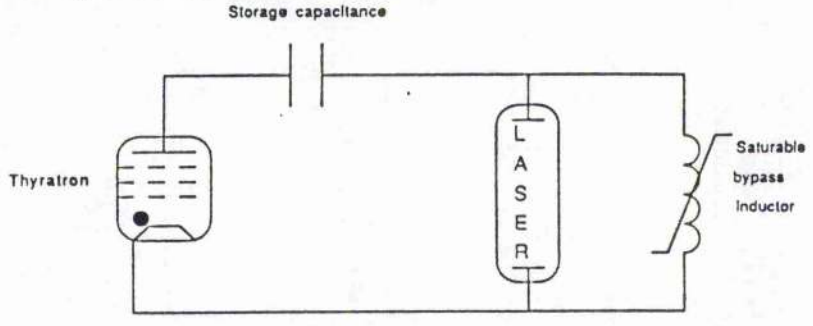


Figure 4a. The Saturable Inductor Modulator Schematic

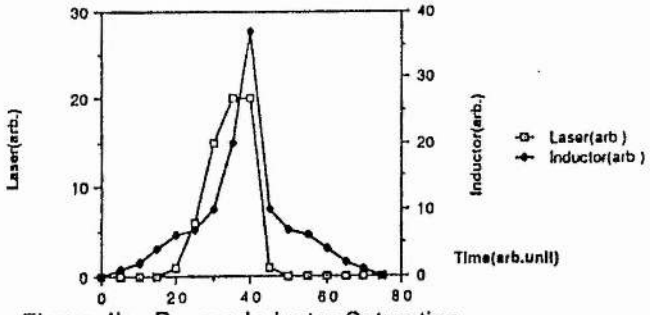


Figure 4b. Bypass Inductor Saturation

The point in time at which saturation occurs can be controlled by biasing the magnetic core.

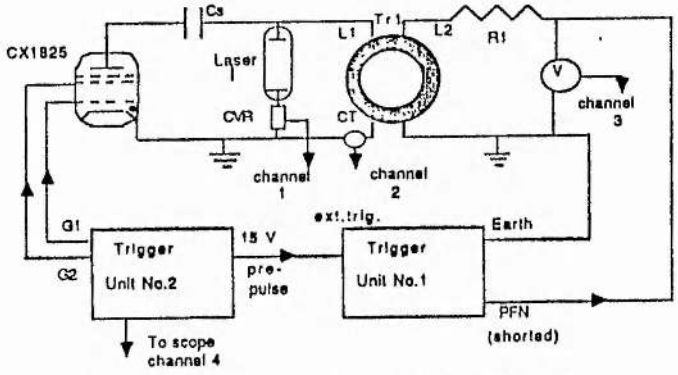


Figure 5a. Pulse biasing of Saturable Inductor

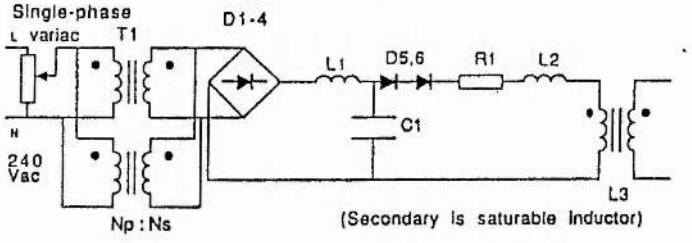


Figure 5b. DC Biasing of Saturable Inductor

Two methods of biasing the magnetic core were investigated. Figure 5a shows the circuit used to bias the core using an auxiliary current pulse, and Figure 5b indicates that saturation may also be controlled using a DC current source.

3. DESIGN OF MAGNETIC SWITCH

Magnetic switching is an established method of transferring stored energy [7,8]. Much of the work on magnetic switching for lasers has been directed at magnetic pulse compression (MPC) [7,8] based on saturable inductors. MPC techniques are primarily used for sharpening the front edge of the current pulse at the load. However, as reported here, saturable inductors may also be used to sharpen the fall-time of the current pulse at the load.

Design Parameters

The unsaturated inductance L_{unsat} of the magnetic switch has the same value as a conventional bypass inductor (approximately 100 μH). The saturated inductance L_{sat} must be minimised. The inductances are given by:-

$$L_{unsat} = \frac{\mu_o \mu_{unsat} N^2 A_c}{l_m} \qquad (1a)$$

$$L_{sat} = \frac{\mu_o \mu_{sat} N^2 A_c}{l_m} \qquad (1b)$$

where N is the number of turns on the core, A_c is the cross-sectional area and l_m is the mean magnetic path length.

The time to saturation of the magnetic switch, t_{sat} , is calculated from the volt-second rating :-

$$V.T = \int_0^{t_{sat}} V_{sw}(t)\, dt = N A_c \Delta B \qquad (2)$$

where $V_{sw}(t)$ is the voltage applied to the magnetic switch and ΔB is the maximum swing in flux density of the core material.

Equations 1(a) and 1(b) show that the ratio L_{unsat} / L_{sat} is simply given by μ_{unsat} / μ_{sat} . For the cores described below, this ratio is of the order of 1000. Equation (2) indicates that the time to saturation increases linearly with the number of turns.

The core properties are summarised in Table 2.

	Toroidal Core No. 1	Toroidal Core No. 2
l_m (m)	0.5	0.25
A_c (cm^2)	3	3.5
ΔB_{max} (T)	0.65	0.8
μ_{unsat}	1000-2000	500-2000
μ_{sat}	< 5	< 5

Table 2. Parameters for the two Magnetic Cores used in the Magnetic Switch.

4. APPARATUS AND RESULTS

Two saturable inductors were constructed using the available ferrite cores, each with an unsaturated inductance of approximately 100 μH (based on the performance of the air-cored inductor).

The first saturable bypass inductor consists of two cores of type No.1 (Table 2) sandwiched together, and wound with ten turns of high-voltage cable (25kV). The second saturable inductor consists of two parallel windings, each of ten turns, wound on a single L4A toroid[9] of type No.2 in Table 2.

Pulse Biasing

The circuit used to pulse bias the core, shown in Figure 5a, operates as follows. Trigger units 1 and 2 are EEV sub-modulators designed primarily for triggering large thyratrons and capable of supplying a 2kV voltage pulse. Trigger unit no. 2, in addition to triggering the thyratron, also sends a 15 volt pre-pulse to trigger unit no. 1. After a variable time-delay, a current pulse is applied to the bias winding of the saturable inductor. With a value of R1 of 100 ohms, a current pulse of 20 A was produced. For the core materials available, saturation occurs with an applied magnetic force of several hundred ampere-turns per metre. Hence, for l_m = 0.5m, and three turns as the bias winding, a 20A current pulse will have a significant effect upon core saturation.

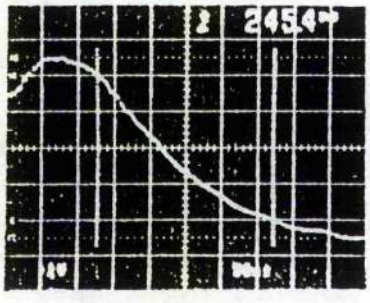
This technique enabled variation of the time to core saturation over the entire duration of the laser current pulse. However, interaction between the circuit components caused distortion of the current pulse applied to the bias winding. Pulse biasing was therefore abandoned in favour of DC biasing, which is described below.

DC Biasing

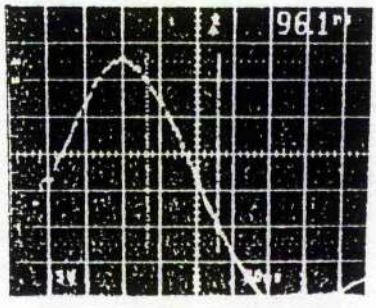
DC biasing provided a more reliable and successful method of controlling the time to saturation of the ferrite core. The circuit shown in Figure 5b can deliver up to 100A into a three turn bias winding. This is sufficient to saturate the core. The L_2/C_2 filter prevents high-voltage spikes induced in the bias winding from damaging the low-voltage circuitry.

Results obtained are presented in Figures 6 and 7, the former being applicable for zero bias current. Figure 6 shows that the ferrite saturates with zero DC bias current at a time shortly after the peak of the laser current pulse. This reduces the laser current fall-time from 275 ns (with the air-cored bypass inductor) to 96 ns. The peak current is 250 amps.

The application of DC bias current allows control over the point in time at which saturation occurs. A DC bias current of 1-2A was sufficient to reduce further the laser current fall-time, to 67 ns, as indicated in Figures 7(i) and 7(ii).

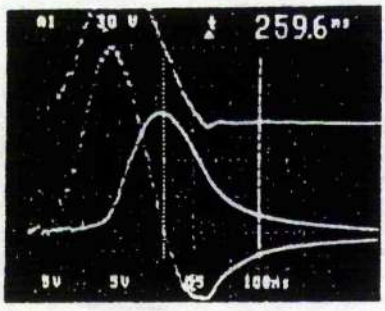


(i) Saturable Inductor Current
Vert : Uncalibrated
Horiz : 50 ns/div

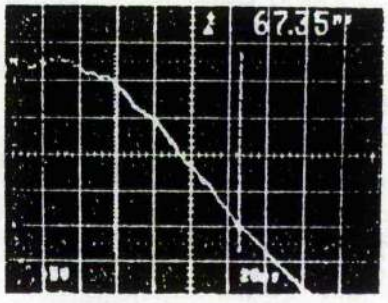


(ii) Laser Current
Vert : Uncalibrated
Horiz : 50ns/div

Figure 6. Current Waveforms for Inductor based on Core no.1 ; Zero Bias Current.



(i) Thyratron (Upper), Laser (Middle) and Saturable Inductor (Lower) Current
Vert : 50A/div
Horiz : 100 ns/div



(ii) Detail of Laser Current

Vert : Uncalibrated
Horiz : 20ns/div

Figure 7. Current Waveforms for Inductor based on Core no.1; Bias Current = $1.1A_{DC}$.

These Figures also reveal that using a parallel saturating bypass inductor increases the current reversal in the laser which is often present in conventional Sr^{++} laser modulator circuits. This reversal has a magnitude of approximately 50A, which is about 20% of the forward current pulse.

5. CONCLUDING REMARKS

The use of a saturable bypass inductor to terminate the current pulse in a recombination laser discharge tube has been demonstrated. The fall-time using an air-cored inductor was 275 ns. This was reduced to 96 ns with zero DC core bias current, and further reduced to 67 ns with 1-2A of DC bias current. Adjustment of the DC bias current provides control over the point at which the core saturates.

For a fixed set of laser operating parameters, DC bias may not be required, since the saturable bypass inductor could be designed to saturate at or near the peak of the laser current pulse.

The presence of the inverse current through the laser provides a source of heating for the free electrons in the discharge. This may adversely affect the laser output as previously discussed. However, a solid-state diode placed in parallel with the laser head as indicated in Figure 8 will provide an alternative low-impedance path for inverse current. This solution has been successfully applied to inhibit current reversal in a conventional Sr^{++} vapour laser circuit[1].

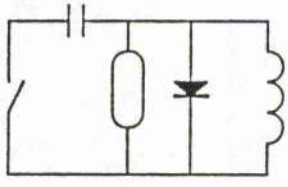


Figure 8. A Method of Inhibiting Current Reversal.

6. ACKNOWLEDGEMENTS

The authors wish to thank Dr. A. Maitland, Mr. E. S. Livingstone and Dr. G. L. Clark for their continued support during this work, and Mrs. Karen Kidd for typing this manuscript. They also wish to thank the directors of EEV for permission to publish this paper.

7. REFERENCES

1. C. E. Little and J. A. Piper, "Development of Efficient High-power Violet Sr^+ and Ultraviolet Ca^+ Recombination Lasers", to be presented at SPIE Symposium on Lasers and Optics, Los Angeles, January 15-20, 1989.

2. A. K. Kidd, "A Compact, Integrated Strontium Vapour Laser System", Ph.D. thesis, University of St. Andrews, St. Andrews, 1989 (to be published).

3. V. V. Zhukov, V. S. Kucherov, E. L. Latush and M. F. Sem, "Recombination Lasers Utilizing Vapours of Chemical Elements. II. Laser Action due to Transitions in Metal Ions", pp. 708-714, Sov. J. Quantum Electron., Vol. 7, No. 6, June, 1977.

4. L. I. Gudzenko, L. A. Shelepin and S. I. Yakovlenko, "Amplification in Recombining Plasmas (Plasma Lasers)", pp. 848-863, Sov. Phys.-Usp., Vol. 17, No.6, may-June, 1975.

5. R. E. Grove, "Copper Vapour Lasers Come of Age", Laser Focus, pp. 45 - 50, July, 1982.

6. J. P. O'Loughlin and J. Sidler, "The End of Line Tail Biter", in conference record of Sixth IEEE Pulsed Power Conference, pp. 692-695, Arlington, Virginia, 1987.

7. I. Smilanski, "Advances in Magnetic Pulse Compression for Copper Vapour Lasers", in conference record of Eighteenth Power Modulator Symposium, pp. 84-85, Hilton Head, South Carolina, 1988.

8. S. E. Ball, "Optimum Switching Time for Magnetic Switches", in conference record of Eighteenth Power Modulator Symposium, Hilton Head, South Carolina, 1988.

9. MEDL Microwave Ferrite Materials, Marconi Electronic Services Limited, Microwave Division, Doddington Road, Lincoln, England. LN6 3LF.

G C A P : A GENERALISED CIRCUIT ANALYSIS PROGRAM FOR PULSED POWER

A. K. Kidd* and A. Maitland

Department of Physics and Astronomy, University of St. Andrews, St. Andrews, Fife, Scotland. KY16 9SS.

* EEV Co. Ltd., 106 Waterhouse Lane, Chelmsford, Essex, England. CM1 2QU.

Abstract

A completely general circuit simulation code based on state variable analysis is described. The code, designated GCAP, has widespread applications to high peak power and repetition rated systems. In essence, GCAP is a pulsed power equivalent of MICROCAP which retains the full range of analysis options and is applicable also to non-linear and time-varying circuits. Furthermore, GCAP contains a library of commonly-used circuit components and facility for the user to define his own component characteristics. Any circuit consisting of non-linear, time-varying circuit elements can be analysed. Output data is in either tabular or graphics form. GCAP is written in standard FORTRAN and may be run on mainframe or on personal computers.

Introduction

In the study of high power pulse generation, system modelling plays an important role both in the design of systems and in producing a quantitative understanding of their performance. Networks are usually designed assuming a constant resistive load. However, power modulators contain variable circuit elements and often supply variable impedance loads. High peak power gas discharge devices have created the need for pulse generators operating into non-linear and time-varying reactive loads with specifications of pulse duration, magnitude, repetition rate, etc. Predictions of system performance by approximate circuit simulation software such as SPICE or MICROCAP suffer from the inability of these programs to model the precise behaviour of continuously changing circuit components such as saturable magnetics, switches and gas discharge loads. Several papers such as [1] have been devoted to this subject. In some cases, the techniques are based on modifying existing codes to simulate actual conditions. More satisfactory, are efforts to develop programs applicable to specific systems [2] but it is precisely this lack of generality which limits interest. Little work appears to have been published on the problem of predicting system performance with variable components.

The circuit simulation code developed aims to increase understanding of thyratron-switched metal vapour laser modulator behaviour and, in particular, to study the interaction of power supply, modulator and the highly non-linear gas discharge loads.

The requirements of the program are as follows.

(1) It should serve as a fast, interactive tool for the study of non-linear, time-varying circuits.

(2) The system should be user-friendly in a manner similar to the commercial models SPICE and MICROCAP.

The resulting program, known as GCAP (Generalised Circuit Analysis Program) satisfies the above requirements. The basis of GCAP is the powerful method of state-variable analysis (SVA). The equations obtained are numerically integrated and the state matrices continuously updated to provide data related to circuit behaviour. GCAP has the ability to accomodate any time-varying and/or non-linear circuit element if it is defined either from a library inherent in the program or by user-specification. This opens up the field of pulsed power to analysis by computer.

This paper describes the program structure. A brief background to the theory and applicability of SVA is presented. GCAP is discussed in terms of the acceptable circuit components, the analysis options available and formatting of input and output data. Finally, an input file for a magnetically-switched copper vapour laser is included as an example.

Computational Technique

GCAP is based on the method of state-variable analysis and considers the internal "state" of a network rather than the external behaviour described by other techniques. In addition to the improved insight into circuit operation that this provides, the method offers the following advantages.

(1) Describing the circuit in terms of SVA results in a set of first order differential equations whose numerical solution is ideally suited to computers. Furthermore, iterative solution of the equations enables both time-varying and non-linear networks to be studied by a simple modification to the A matrix.

(2) Matrix inversion yields results in the frequency domain which are valuable in discussing aspects of control system design such as bandwidth, gain and phase margins and corner frequencies.

Theoretical Background

Classical analysis of second-order circuits consisting of linear time-invariant elements is based on either nodal or mesh methods and leads in general to a set of coupled second order scalar differential equations. However, this approach, which is used by the circuit simulators MICROCAP and SPICE, does have inherent limitations with regards to both non-linear and time-varying systems which restrict its range of application, particularly in the field of pulsed power.

The technique of state-variable analysis (SVA) relies upon the fact that any scalar differential equation of order n can be expressed as a minimal set of n first order differential equations. In state-space form, these equations are

$$\dot{x}(t) = Ax(t) + Bu(t) \quad \text{STATE EQUATION} \quad (1)$$

and

$$y(t) = Cx(t) + Du(t) + E\dot{u}(t). \quad \text{INPUT-STATE-OUTPUT EQUATION} \quad (2)$$

In these equations, $x(t)$, $u(t)$ and $y(t)$ are, respectively, the state, input and ouput vectors, $x = (x_1, x_2, \ldots, x_n)^T$ is a real variable) and A to E are matrices. The state variables $x_1, \ldots, x_n$ are the independent initial conditions which the system can support. The state model (equations (1) and (2)) completely determines the time-evolution of the system. Since the q-v and ϕ-i characteristics of time-varying and non-linear elements are not constant, it is suitable to choose the state vector as the set of all capacitance charges and all inductance fluxes. Thus we have

$$x(t) = \begin{pmatrix} q_C(t) \\ \phi_L(t) \end{pmatrix} . \quad (3)$$

where C(t) and L(t) are, in general, the small-signal capacitance and inductance at the operating points V_C and I_L, and are prescribed functions of time.

Numerical solution of the differential state equations (1) in the time-domain is obtained by means of the Runge-Kutta algorithm. This iterative approach to solving the state equations provides SVA with one of its most powerful features. By updating the state matrices at each successive time step, non-linear and time-varying characteristics can be accounted for.

Program Description

GCAP is a general-purpose circuit simulation program. It is written in standard FORTRAN and may be run on mainframe or personal computers. The present version of GCAP contains approximately 10,000 executable statements and occupies about 150K of RAM.

The program can cope with almost any component including those which may be non-linear, time-varying, and voltage- or current-sensitive. Thyratron switches, spark gaps and laser loads are examples of elements modelled by GCAP that are beyond the scope of MICROCAP.

The user interface is simple and straightforward. Interaction with GCAP is by means of three commands, namely component, analysis and structure descriptors. A descriptor is simply an input statement, the exact format of which is indicated in the example to follow. Descriptors are entered in the form of input files in any order and at any stage when running the program.

(1) Components

The user of GCAP need only specify circuit element type and value, nodal connections and relevant non-linear/time-varying data to completely describe the circuit. Component descriptors are used for this purpose. Data is input on two separate lines, the first containing character data (to identify the element and its type) and the second numerical data (element value, nodal connections and any non-linear/time-varying parameters). To maintain generality, the user requires no special knowledge of the computational technique to enter this information. Numbers to designate nodes must be positive integers but need not be consecutive. An algorithm is included to classify the fixed topology network branches into a proper tree. By means of successive matrix manipulations, explicit expressions are obtained for the state matrices A to E in Equations (1) and (2).

Examples of circuit components which come within the scope of GCAP are listed in Table 1.

Element Type	Examples
Passive components	Resistors, capacitors, inductors, independent and controlled voltage and current sources.
GCAP library	Transmission lines, diodes, BJT's, MOSFET's, SCR's.
User-defined components	Switches, time-varying sources, flux-controlled magnetic components, voltage-controlled gas discharge devices.

Table 1. Acceptable components

Modelling of Components. Library models are sub-circuits which, in GCAP, are considered as single elements. The user needs only to specify the relevant model parameter values to define the element. User-defined models are also entered as sub-circuits. This feature allows simulation of switches, saturable magnetic devices, and the like.

The time-dependent behaviour of components is entered explicitly by the user. For the purpose of GCAP, opening and closing times of switches are divided into chosen intervals. New state matrices are formulated at each successive time step. For continuity, the final switch states at any one time are the initial states at the next time step.

The parameters used to control non-linear elements are listed in Table 2. One of the two distinct state variables must be chosen for each element.

Element	Control Parameter (State variable)
Capacitor	Voltage, V_c Charge, q_c
Magnetics (inductors, transformers)	Current, I_L Flux, ϕ_L
Switches	Voltage, V Current, I

Table 2. Control parameters for non-linear components

(2) Analyses

Analysis options may be expressed in terms of four categories and specified to GCAP at any stage during the program execution. The categories are DC, which generally aims to calculate the DC operating point of the circuit; AC, which examines the circuit response to sinusoidal input; the so-called transient category in which temporal behaviour of the circuit is calculated in the time-domain, naturally; and the final category is "frequency", which is used for frequency domain calculations. As an example of a transient analysis, the system response can be separated into the forced (zero-input or steady-state) and natural (zero-state or transient) responses, and studied individually. The circuit as a whole, or any subcircuit (eg a semiconductor device or any specified group of elements) can be analysed for its response to a transient voltage pulse. The rise-time, fall-time, overshoot, etc. of the resulting waveform can be calculated.

Each analysis contains further sub-divisions. For example, GCAP may be requested to perform a sensitivity analysis in which the values of all components are systematically varied by either a user-specified amount, or 5% (default value), and the circuit studied to see whether it performs within specification limits. Also, the effect of different temperatures on performance can be investigated.

A more detailed description of programs as they are applied to various circuits is given in the GCAP user manual [3].

(3) Structure Descriptors

The third and final set of commands are structure descriptors which are general statements describing the formatting of the I/O data. These include the PRINT and PLOT commands, which are required for tabular listings of data and graphics, respectively. The user can specify which parameters are to be displayed. The voltage across, current through or power dissipated/energy stored in any circuit element can be monitored and printed over a chosen time interval. Alternatively, a graphics subroutine provides a visual display of the results.

Any additional output data such as comments, headings, etc., are classed as structure descriptors.

Program Flow Diagram

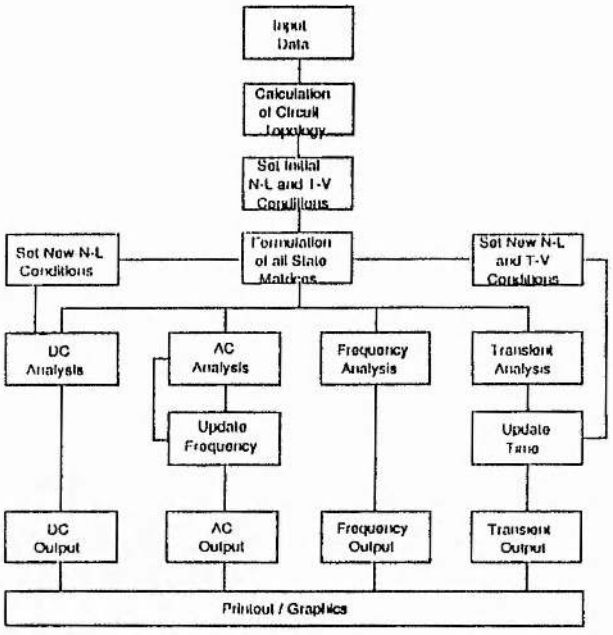


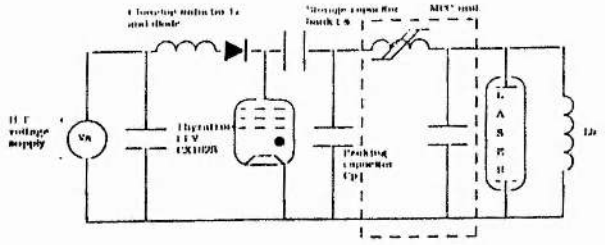
Figure 1. GCAP program flow diagram

Example Input File:
Simulation of Magnetic Pulse Compression

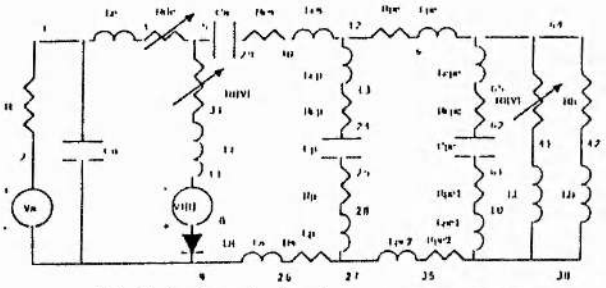
An application of GCAP is illustrated in the analysis of a pulsed gas discharge laser circuit with one stage of magnetic pulse compression (MPC) (Figure 2a). The active medium may be either copper vapour [4] or an excimer [5]. The lumped parameter circuit to be analysed by GCAP is shown in Figure 2b and contains three variable components - the flux-dependent saturable inductance, the time-varying thyratron impedance and the voltage-controlled laser impedance.

Table 3 shows a section of the input file to be entered by the user. This file requests GCAP to perform a transient analysis of the circuit and plot the laser current, the energy stored in the saturable

inductor and the thyratron voltage over time periods of 500ns and 5μs, respectively. Finally, the effect of varying the unsaturated inductance of the magnetic switch on each of these quantities is studied.



(a) Ideal circuit



(b) Equivalent circuit to be analysed by GCAP

Figure 2. Copper vapour laser circuit with one stage of MPC

```
TITLE    MAGNETIC PULSE COMPRESSION

R  RCP
3m2   13   24

R  RLV  VAR
800   41   64   50   1k2

C  CO
1U   1    9

:    :    :

TRAN

TIME
1N   5U

PLOT  I   RLV
0   500N  25N

PLOT  E   LPC
0   500N  25N

PRINT   V   RTV
0   5U  500N

SENSE   LPC
100U    500U    100U

END
```

Table 3. Input file applicable to the magnetic modulator circuit of Figure 2b

Conclusions

A general circuit analysis program has been described which can be applied to non-linear circuits and a wide range of pulsed power problems. The code requires no specialised knowledge of the computational method and may provide DC, AC, transient and frequency solutions.

Acknowledgements

The authors wish to thank the Directors of EEV for permission to publish this paper.

References

[1] C. Young, "Modeling pulsed electric discharges using a circuit analysis code", In Proceedings of IEEE Sixth International Pulse Power Conference, 1987, pp. 397-400.

[2] R. M. Roark, M. E. Parten, L. B. Masten and T. R. Burkes, "Pulse forming networks with time-varying or non-linear resistive loads", In Proceedings of IEEE Thirteenth Power Modulator Symposium, 1978, pp. 46-51.

[3] A. K. Kidd, "GCAP Version 01 User Manual", Reference Manual, University of St. Andrews (1989).

[4] R. A. Petr, J. F. Zumdieck, J. Demboski, I. Smilanski, J. J. Ewing and R. E. Center, "Magnetic pulse compression for copper-vapour lasers", In Proceedings of IEEE Fourth International Pulse Power Conference, 1983, pp. 236-241 (IEEE Cat. 83CH1908-3).

[5] I. Smilanski, S. R. Byron and T. R. Burkes, "Electrical excitation of an XeCl laser using magnetic pulse compression", Appl. Phys. Lett., vol. 40(7), pp.547-548, April 1982.

COMPUTER MODELLING OF METAL VAPOUR LASER MODULATOR CIRCUITS

A. K. Kidd and A. Maitland

Department of Physics and Astronomy, University of St. Andrews, St. Andrews, Fife, Scotland. KY16 9SS.

Introduction

Rare-gas halide and a number of metal vapour lasers require pulsed electrical excitation. In the case of the copper vapour laser (CVL) [1], this is because the lower laser level is a metastable state and must be allowed to relax during the interpulse period. The strontium vapour laser (SVL) [2] is a recombination laser in which lasing is observed in the discharge current afterglow. It is well-known that all of these lasers experience impedance matching problems between modulator and gas discharge load; the inverse voltages and currents produced may be as high as 50% of the peak forward values. The thyratron switch may be destroyed if the arc produced damages the cathode surface. This has led to the development of the reverse-conducting hollow anode thyratron [3]. Described here is a computer model of both the copper and strontium laser circuits in an effort to improve the matching and reduce the strain on the thyratron switch. The difference in performance between the two modulators will be examined.

The Model

The resistance of the SVL plasma R_p varies by several orders of magnitude over the period of one discharge cycle. It is a complex function of free electron density n_e and temperature T_e, aswell as strontium level populations N_i:-

$$R_p = \frac{3kT_e \sum_i N_i \sigma_{mi}}{n_e e^2 \langle v_e \rangle A} \qquad (1)$$

If the current pulse applied to the SVL is rapidly terminated, the electron temperature falls off exponentially until "stationary drainage" [4] is established, where T_e is essentially constant. This situation arises because the characteristic relaxation times of free electron density greatly exceed the average time of collisional de-excitation between levels. During this time, the net flow of particles into a given atomic/ionic state is zero. i.e.

$$\frac{\delta n_e}{\delta t} = \frac{\delta N_{i>1}}{\delta t} = 0 \qquad (2)$$

and the resistance of the plasma is essentially constant.

Kinetic coding [4,5] reveals that a time scale of 10^{-9} to 10^{-8} seconds is required to achieve this condition. Thus, if the current pulse fall-time is sufficiently short, the resistance of the plasma will rapidly achieve a constant value. If the circuit components are chosen such that this resistance is maintained for a time greater than the current pulse duration then it should be possible to impedance match the modulator and discharge load.

The Model

The CVL/SVL modulator circuit is shown in Figure 1a. The storage capacitance C_s is resonantly charged from a DC power supply via a charging inductor L_c and a bypass inductor L_b. C_p is a peaking capacitor which is only present in the CVL circuit.

Figure 1b shows the lumped parameter circuit used in the computer model. Due to the limitations of SPICE and MICROCAP, it was necessary to develop a simulation code to model time-varying and non-linear components. The thyratron is treated as a unidirectional device; this is equivalent to a series-connection of a binary resistance, fixed inductance and diode. The switching action is represented by an exponentially-decaying reverse-biased voltage source which is removed after a time τ_t (to represent the finite thyratron turn-on time). The laser head is also modelled by a series-connection of a binary resistance with a turn-on time of τ_l and fixed inductance, but now current flow is bidirectional. The inductances L_s and L_p represent lumped circuit values.

Results

The equations describing the circuit are iterated using a Runge-Kutta algorithm on a sub-picosecond time-scale to simulate circuit behaviour.

Figure 2a shows the predicted laser, bypass

inductor and thyratron current waveforms for the SVL. The storage capacitance charging voltage is 10 kV and the component values are as indicated. The same currents are shown in figure 2b on an expanded scale. Once the gas is broken down, the SVL presents a low impedance path. As the thyratron is unidirectional, the energy stored in the bypass inductor is dissipated by a reverse current through the laser. This appears as an overshoot current whose decay time-constant is $(L_b+L_l)/(R_b+R_l)$ (Figure 2b). Figure 3 shows the corresponding voltage across the laser head.

Figure 4 illustrates the possibility of removing this overshoot current by critically matching the laser load to the modulator. This is achieved for a plasma impedance of about 20-30Ω.

Figure 5 shows the corresponding voltage across the laser head for R_l = 30 Ω.

In the case of the CVL the situation is made more complicated by the presence of the peaking capacitor (Figure 1a). The laser current now exhibits a double peak. Computer modelling of the CVL [6] shows that the plasma resistance may vary by one or two orders of magnitude between 200 and 400 ns after the start of the discharge. This is after the current pulse has ended and hence matching is not possible.

References

[1] R.S.Grove, "Copper Vapour Lasers Come of Age", Laser Focus, pp.45-50, July, 1982.

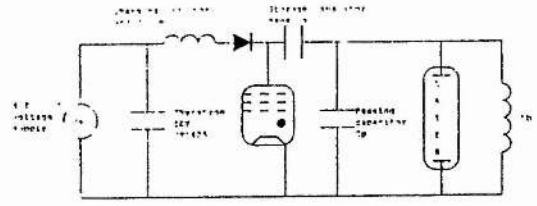
[2] C.W.McLucas and A.I.McIntosh, "Discharge heated longitudinal Sr^+ recombination laser", J.Phys.D. 13, pp.1189-1195, 1986.

[3] H.Menown and C.V.Neale, "Thyratrons for Short Pulse Laser Circuits", Thirteenth Power Modulator Symposium, June, 1978.

[4] D.R.Bates, A.E.Kingston and R.W.P.McWhirter, "Recombination between electrons and atomic ions.I.Optically thin plasmas",Proc.Roy.Soc., A267 p.297, 1962.

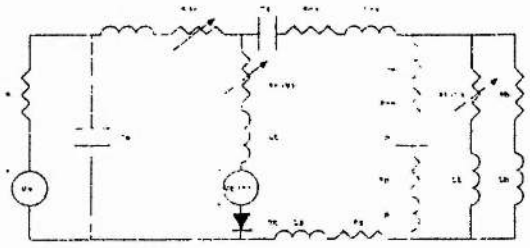
[5] A.K.Kidd, "A Compact, Integrated Strontium Vapour Laser System", Ph.D. thesis, University of St. Andrews, St. Andrews, 1989 (to be published).

[6] G.L.Clark, private communication.



(a) Ideal

Figure 1. Modulator circuits used in the numerical analysis



(b) The Computer Model

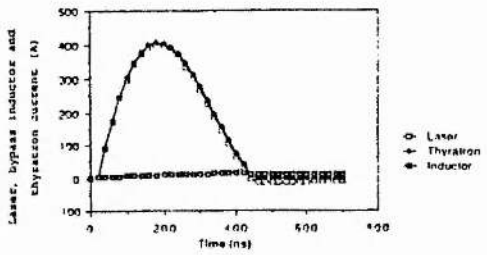


Figure 2a. Predicted current waveforms for the SVL. C_s=8 nF; R_l=10 ohms; L_b= 0.1 mH

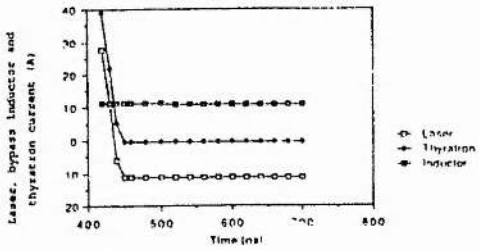


Figure 2b. As Figure 2a but current scale expanded

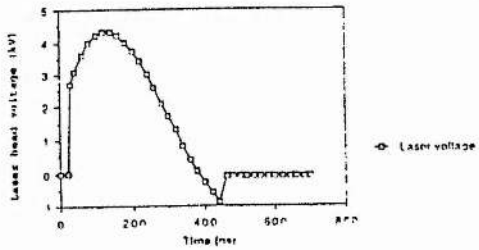


Figure 3. Laser voltage

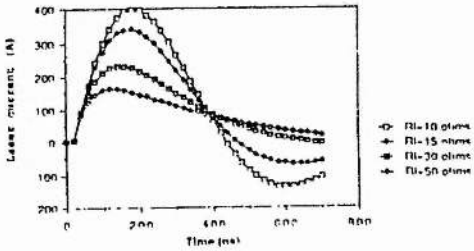


Figure 4. Variation of laser current with plasma resistance

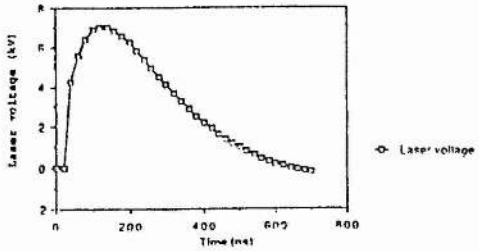


Figure 5. Voltage across the laser head for a plasma impedance of 30 ohms

THE APPLICATION OF STATE VARIABLES TO THE SIMULTANEOUS ANALYSIS OF LASER CIRCUITS AND GAS DISCHARGE POPULATIONS

A. K. Kidd and A. Maitland

Department of Physics and Astronomy, University of St. Andrews,

St. Andrews, Fife, Scotland. KY16 9SS.

Introduction

One of the most interesting aspects of state-variable analysis (SVA) is that cases consisting of a wide range of physical systems which are interconnected but which operate according to a different set of laws can be analysed simultaneously ie the system can be viewed as one mathematical model. In this paper, we show that SVA can be applied to analyse a pulsed gas discharge laser system in which atomic, thermodynamic and electrical state equations are considered together.

The Method of State-variable Analysis

Presented here is a novel technique of describing phenomena in electrically excited gaseous systems. In a gaseous discharge of cylindrical symmetry and with no assumption of constant particle temperatures, the coefficients in the particle conservation law exhibit a radial dependence. The simultaneous solution of coupled electrical, atomic and thermodynamic equations with variable coefficients is a problem outside the scope of circuital methods such as Laplace transforms. However, application of the method of state-variable analysis enables both time-varying and non-linear processes to be studied.

State variables describe the internal states of the system. There may exist m atomic (number density of particles in a particular state, free electron density, number of photons), n thermodynamic (local particle trmperatures and pressures, energy densities of the gas and electrons) and p electrical (capacitance charges and inductance fluxes) state variables. These are the independent initial conditions which the system can support. If the state variables are designated N^m, T^n and E^p respectively then the system can be described by coupled rate equations describing the time-evolution of these quantities:

$$
\begin{aligned}
N_1 &= a_{1,1}N_1 + \ldots + a_{1,m}N_m + a_{1,m+1}T_1 + \ldots + a_{1,m+n}T_n + a_{1,m+n+1}E_1 + \ldots + a_{1,m+n+p}E_p \\
&\vdots \\
N_m &= a_{m,1}N_1 + \ldots + a_{m,m}N_m + a_{m,m+1}T_1 + \ldots + a_{m,m+n}T_n + a_{m,m+n+1}E_1 + \ldots + a_{m,m+n+p}E_p \\
\dot{T}_1 &= a_{m+1,1}N_1 + \ldots + a_{m+1,m}N_m + a_{m+1,m+1}T_1 + \ldots + a_{m+1,m+n}T_n + a_{m+1,m+n+1}E_1 + \ldots + a_{m+1,m+n+p}E_p \\
&\vdots \\
T_n &= a_{m+n,1}N_1 + \ldots + a_{m+n,m}N_m + a_{m+n,m+1}T_1 + \ldots + a_{m+n,m+n}T_n + a_{m+n,m+n+1}E_1 + \ldots + a_{m+n,m+n+p}E_p \\
\dot{E}_1 &= a_{m+n+1,1}N_1 + \ldots + a_{m+n+1,m}N_m + a_{m+n+1,m+1}T_1 + \ldots + a_{m+n+1,m+n}T_n + a_{m+n+1,m+n+1}E_1 + \ldots + a_{m+n+1,m+n+p}E_p \\
&\vdots \\
\dot{E}_p &= a_{m+n+p,1}N_1 + \ldots + a_{m+n+p,m}N_m + a_{m+n+p,m+1}T_1 + \ldots + a_{m+n+p,m+n}T_n + a_{m+n-p,m+n+1}E_1 + \ldots + a_{m+n+p,m+n+p}E_p
\end{aligned}
$$

or

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \qquad (1)$$

where A and B are matrices and u(t) is a vector of sources of excitations. A is a square matrix of dimension (m+n+p).

Numerical solution of the state equations is by means of the Runge-Kutta algorithm. The characterisation of the entire system by a set of coupled first order differential equations provides SVA with one of its most advantageous features. By updating the state matrices at each successive time step, non-linear and time-varying characteristics can be included. Furthermore, state-variable analysis provides a powerful method of manipulating the results obtained by analysis of the A matrix.

Generalised Circuit Analysis Program

In practice, the system of equations (1) may prove difficult, if not impossible, to formulate by inspection. However, the system can be analysed by means of the Generalised Circuit Analysis Program (GCAP) [1,2]. This is a general-purpose circuit simulation program written in standard FORTRAN which may be run on mainframe or personal computers. Lumped parameter circuits are used in the computer model. In the case of modeling a complex circuit component such as a gas discharge load, the user defines the element by means of the usual rate equations and higher order moments of the Boltzmann equation. Given appropriate specified initial conditions, a subroutine proceeds to calculate the equivalent instantaneous resistance and inductance of the plasma. The former is a complex function of free electron density n_e and temperature T_e, as well as strontium level populations N_i

$$R_p = \frac{3kT_e\sum_i N_i\sigma_{mi}}{n_e e^2 \langle v_e \rangle A} \qquad (2)$$

and the latter is given by

$$L_p = \frac{m_e d}{n_e e^2 A} \qquad (3)$$

These parameters are then incorporated into the circuit to be analysed by GCAP.

Component Modelling

GCAP contains a library of complex circuit components, or the user may define elements to the code. The element must be represented by an electrical analogue, such as describing gas discharge devices in terms of plasma impedance. The user must define the characteristics of a device to GCAP by means of a subroutine. GCAP can accomodate any circuit consisting of non-linear, time-varying elements. This facility allows treatment of devices such as the thyratron, spark gap and laser load, inconjunction with saturable magnetics, etc.

The parameters used to control non-linear elements are listed in Table 1.

Element	Control State Variable
Capacitor	Voltage, V_c
	Charge, q_c
Magnetics	Current, I_L
	Flux, ϕ_L
Switches	Voltage, V
	Current, I

Table 1. Control parameters for non-linear elements

An example of the use of GCAP is described below.

Example. Unified State Variable Analysis of Pulsed Gas Discharge Loads

The strontium vapour laser requires pulsed electrical excitation. A typical discharge circuit [3] would be defined to GCAP by means of the input Table 2 below.

```
TITLE      STRONTIUM VAPOUR LASER

C     C0
1U    4     6
:     :     :
R     RL
SUBA

L     LL
SUBB

TRAN

TIME
1N    5U

PLOT I      RL
0    500N   25N

PRINT       V     RT
0     5U    500N

END
```

Table 2. Input file applicable to the circuit ofFigure 1b [3]

SUBA and SUBB represent subroutines entered by the user to define the atomic and thermodynamic state variables to GCAP, which in turn define the plasma resistance and inductance. This file requests GCAP to perform a transient analysis of the circuit and plot the laser current and thyratron voltage over time periods of 500ns and 5μs, respectively. The output data may be in either tabular or graphics form.

Conclusions

A general circuit analysis program has been described which can accomodate non-linear and time-varying devices and which can be applied to a wide range of gaseous electronics problems. Thyratron switches, spark gaps and laser loads are examples of devices which may be analysed by GCAP which are beyond the scope of commercially available circuit analysis programs such as MICROCAP or SPICE. This opens up the field of gas discharge devices to analysis by computer.

References

[1] A.K.Kidd and A. Maitland, "GCAP : A generalised circuit analysis program for pulsed power", in Proceedings of IEEE Seventh International Pulsed Power Conference, Monterey, 1989.

[2] A. K. Kidd, "GCAP Version 01 User Manual", Reference Manual, University of St. Andrews (1989).

[3] A. K. Kidd and A. Maitland, "Computer modelling of metal vapour laser modulator circuits", to be presented at ICPIG XIX, Belgrade, Yugoslavia (1989).

Appendix D

Strontium Vapour Laser Model Program Listing

1. INTRODUCTION

SRRAD is a homogeneous state-space model for the simulation of a pulsed electrical discharge in a helium buffer gas-strontium vapour mixture. The code is employed in this thesis to examine the effects of parametric variation of the modulator circuit (storage capacitance, charging voltage, pulse repetition frequency) on microscopic parameters such as species densities and particle temperatures. Although the code is written to simulate strontium laser performance, it is readily adapted to apply to any repetitively pulsed recombination laser system by modification of the relevant atomic constants.

2. SCREEN EDITING OF INPUT PARAMETERS

Due to the large number of variables which may be input to SRRAD, the program contains default values. The default parameters relevant to the SVL model described in this thesis are listed in Table 4.1. A screen editor allows the user to display or edit the input values and these are illustrated in the first section of this appendix. A subprogram provides an on-screen display of the main simulation parameters and the number of time-steps executed. The format of the output data has been modified to display the instantaneous values of the electrical state variables (voltages and currents) and the circuit elements (resistances, capacitances and inductances) as the program proceeds.

3. SRRAD PROGRAM LISTING

A complete program listing of SRRAD is included in this appendix to enable the user to reproduce the results reported in this thesis or extend the parameter space investigated. Furthermore, the user may modify or extend the source code as required.

The program structure is illustrated in the flow diagram of Figure 4.2. The main program is known as SRRAD. The functions of the subprograms are as follows:

SRINIT:	Initialisation of input parameters of current SVL system
SRHEAD:	On-screen display of parameters in use
SRELEC:	Calculation of plasma resistance and inductance
SRTEMP:	Calculation of particle temperatures and pressures
SRRATE:	Calculation of plasma kinetic properties - rate constants and population densities
SROPTO:	Calculation of small-signal gain and absorption coefficients, laser intensity.

==========

KINETIC MODELLING OF He-Sr RECOMBINATION LASERS

==========

SRRAD is a two-dimensional SVL simulation
program designed to investigate the kinetic processes
occuring within the active medium

Title of simulation:
SVL Cavity Gain and Loss

SVL Cavity Gain and Loss: ANALYSIS LIMITS

A....EDIT/REVIEW PARAMETERS (Y/N):		N
B....MAXIMUM SIMULATION TIME (ns):		3010.00
C....TIME INCREMENT (ps):		20.00
D....UPPER TRACE A:		1
E....UPPER TRACE B:		2
F....UPPER TRACE RANGE (HIGH/LOW):		
G....LOWER TRACE A:	0.00	0.00
H....LOWER TRACE B:		3
I....LOWER TRACE RANGE (HIGH/LOW):		4
J....CRT PLOT (GRAPH) OR TABLE (TABLE):	0.00	TABLE
K....SAVE DATA TO USER FILE (Y/N):		N
L....RADIAL RESOLUTION REQUIRED (10/20/50 OFF-AXIS POINTS):		0

Hit menu to edit a value, 1 to edit a parameter, 2 to run, 3 to quit

SVL Cavity Gain and Loss: CONFIGURATION OF LASER HEAD

Q....DISCHARGE TUBE WIDTH (cm):	1.30
R....DISCHARGE TUBE DEPTH (cm):	1.30
S....DISCHARGE TUBE LENGTH (ELECTRODE SEPARATION) (cm):	50.00
T....LENGTH OF OPTICAL CAVITY (MIRROR SEPARATION) (cm):	150.00
U....OUTPUT MIRROR REFLECTIVITY (fraction):	0.30

Hit menu to edit a value, 1 to view next page, 2 to run, 3 to quit

SVL Cavity Gain and Loss: ATOMIC PARAMETERS

W....INITIAL AXIAL ELECTRON DENSITY (cm-3):	4.00E+0012

Hit menu to edit a value, 1 to view next page, 2 to run, 3 to quit

SVL Cavity Gain and Loss: THERMODYNAMIC PARAMETERS

a....TOTAL EXTERNAL GAS PRESSURE (mbar):	200.00
b....DISCHARGE TUBE INNER WALL TEMPERATURE (degress K):	900.00
c....PERCENTAGE OF HELIUM IN GAS MIX (%):	100.00
d....PERCENTAGE OF STRONTIUM IN GAS MIX (%):	0.00
e....PERCENTAGE OF NEON IN GAS MIX (%):	0.00

Hit menu to edit a value, 1 to view next page, 2 to run, 3 to quit

SVL Cavity Gain and Loss: ELECTRICAL PARAMETERS

g....POWER SUPPLY VOLTAGE (kV):	7.20
h....STORAGE CAPACITANCE (nF):	4.00
i....PEAKING CAPACITANCE (nF):	0.00
j....CHARGING INDUCTANCE (mH):	150.00
k....UNSATURATED MPC INDUCTANCE (microhenries):	100.00
l....RELATIVE PERMEABILITY OF FERRITE:	500.00
m....SATURATION CURRENT (amps):	75.00
n....BYPASS ELEMENT RESISTANCE (ohms):	5.00
o....BYPASS ELEMENT INDUCTANCE (microhenries):	100.00
p....DISCHARGE LOOP RESISTANCE (ohms):	1.00
q....DISCHARGE LOOP INDUCTANCE (nanohenries):	1.00

Hit menu to edit a value, 1 to view next page, 2 to run, 3 to quit

SVL Cavity Gain and Loss: LASER AND THYRATRON PARAMETERS

t....LASER HEAD RESISTANCE (ohms):	1.00
u....LASER HEAD INDUCTANCE (nH):	500.00
v....THYRATRON OFF-RESISTANCE (ohms):	100000.00
w....ARC-DROP THYRATRON RESISTANCE (ohms):	2.00
x....THYRATRON INDUCTANCE (nH):	500.00
y....THYRATRON TURN-ON TIME (ns):	5.00

Hit menu to edit a value, 2 to run, or 3 to quit

time = 0 nsec:
I = 0.00E+0000 DIT = 0.00E+0000DST = 0.00E+0000
NE = 4.00E+0012 TE = 9.00E+0002 DTET = 0.00E+0000
S1 = 2.42E+0014 S2 = 4.00E+0012 S3 = 0.00E+0000 S4 = 0.00E+0000
S5 = 0.00E+0000 S6 = 0.00E+0000 S7 = 0.00E+0000
S8 = 0.00E+0000 S9 = 0.00E+0000 S0 = 0.00E+0000
H1 = 1.61E+0018 H2 = 0.00E+0000 H3 = 0.00E+0000 H4 = 0.00E+0000
H5 = 0.00E+0000 H6 = 0.00E+0000
N1 = 0.00E+0000 N2 = 0.00E+0000 N3 = 0.00E+0000 N4 = 0.00E+0000
N5 = 0.00E+0000 N6 = 0.00E+0000
P430 = 0.00E+0000 P416 = 0.00E+0000
E-plasma = 0.00E+0000 V_plasma = 0.00E+0000
TG = 9.00E+0002 DTGT = 0.00E+0000
DH2T = 0.00E+0000 DH3T = 0.00E+0000 DN3T = 0.00E+0000
R35 = 0.00E+0000 R38 = 0.00E+0000
R42 = 0.00E+0000 R43 = 0.00E+0000 R48 = 0.00E+0000
R77 = 0.00E+0000 R80 = 0.00E+0000
R84 = 0.00E+0000 R85 = 0.00E+0000 R90 = 0.00E+0000
S_plasma = 3.22E-0001 R_plasma = 1.17E+0004 L_plasma = 3.34E-0008
21 = 0.00E+0000 109 = 0.00E+0000 113 = 0.00E+0000 151 = 0.00E+0000
6 = 0.00E+0000 7 = 0.00E+0000 17 = 0.00E+0000
time = 0.00E+0000 ns
Voltage across storage capacitor = 1.44E+0004 V
Voltage across thyratron = 1.44E+0004 V
Voltage across laser head = 0.00E+0000 V
Current in laser head = 2.78E-0001 A
Current in thyratron = 2.79E-0001 A
Current in bypass element = 1.48E-0003 A
Resistance of laser head = 1.17E+0004 ohms
Resistance of thyratron = 1.00E+0005 ohms
Resistance of bypass element = 5.00E+0000 ohms
Inductance of laser head = 5.33E-0007 henries
Inductance of MPC switch = 1.00E-0004 henries
Inductance of thyratron = 5.00E-0007 henries
Capacitance of storage capacitor = 4.00E-0009 nanofarads

DISCHARGE TUBE WIDTH (cm): 1.3

DISCHARGE TUBE DEPTH (cm): 1.3

DISCHARGE TUBE LENGTH (ELECTRODE SEPARATION) (cm): 50.0

INITIAL AXIAL ELECTRON DENSITY (cm-3): 4.0E+0012

TOTAL EXTERNAL GAS PRESSURE (mbar): 200.0

PERCENTAGE OF HELIUM IN GAS MIX (%): 100.0

PERCENTAGE OF NEON IN GAS MIX (%): 0.0

POWER SUPPLY VOLTAGE (kV): 7.2E+0000

STORAGE CAPACITANCE (nF): 4.0

Run in progress with the above values: 150500 steps

Time-step 1

```
{$B+}    {Boolean complete evaluation on}
{$S+}    {Stack checking on}
{$I+}    {I/O checking on}
{$N+}    {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}

PROGRAM SRRAD0;          {Version 1    Created on 17th October 1990}
                         {Last modified on 18th May 1991}

{$R+}    {Compiler range checking for arrays, etc.}
{$F+}    {Far call compiler directive}

{
 SRRAD is a third-generation, zero-dimensional strontium vapour laser
 simulation.  This is version 2 of the program, which includes
 specific reference to the laser system in operation within the Department of
 Physics and Astronomy at the University of St. Andrews, St. Andrews, Fife,
 Scotland.  The program takes account of the effect of adding different
 proportions of neon to the helium-strontium mixture.

 The model considers kinetics within a cylindrically symmetric zero-dimensional
 plasma.

 The input routine has been devised to act like a screen editor with preset
 default values for the current SVL system configuration.  The format
 of the output data has been written to enable the display of laser
 intensities, discharge voltage and current waveforms and circuit component
 values.

 Also included is the option of iterating various input parameters over a
 user-specified range of values.
 }

{Start of global type, constant and variable declarations}

Uses
  Overlay,
  Crt,
  Printer,
  SrInit0,
  SrHead0,
  SrRate0,
  SrElec0,
  SrTemp0,
  SrOpto0;
{ SrEqns }

{$O SrInit0}
{$O SrHead0}
{$O SrElec0}
{$O SrRate0}
{$O SrTemp0}
{$O SrOpto0}
```

```
{-------------------------------------------------------------------------}

{Now declare the procedures and functions}

PROCEDURE INSTALL;

{Initialisation of Turbo Pascal's Overlay Manager}

const     OvrMaxSize = 8;

var       OvrName:          string[79];
          Size:             LongInt;

begin {Start of initialisation of Turbo Pascal's Overlay Manager}

    OvrName:='SRRAD0.OVR';
    repeat
       OvrInit (OvrName);
       if OvrResult=ovrNotFound then
       begin
          writeln('Overlay file not found: ',OvrName,'.');
          write('Enter correct overlay file name: ');
          readln(OvrName);
       end;
    until OvrResult<>ovrNotFound;

    if OvrResult<>ovrOK then
    begin
       writeln('Overlay manager error.');
       Halt(1);
    end;
    OvrInitEMS;
    if OvrResult<>OvrOK then
    begin
       case OvrResult of
       ovrIOError:       write('Overlay file I/O error');
       ovrNoEMSDriver: write('EMS driver not installed');
       ovrNoEMSMemory: write('Not enough EMS memory');
    end;
    write('.Press Enter...');
    readln;
    end;
    OvrSetBuf(OvrMaxSize);
end;      {End of initialisation of Turbo Pascal's Overlay Manager}


PROCEDURE CONSTANT;

{
Procedure to calculate the general constants used in the model
}

begin
```

```
{   N := round(sim_time*1000/inc_time);   } {Number of steps in calculation}
    AR := (pi/4)*amp_width*amp_depth;      {Cross-sectional area of
                                            discharge}
{   pump_vol := amp_length*AR;  }          {Total amplifier pumped volume}

    CL1 := 0.347;                          {Coulomb logarithm for z = 1}
    CL2 := 0.805;                          {Coulomb logarithm for z = 2}
    Ry := 13.6;                            {Rydberg energy}

{Short form constants}

        AQ := sqrt((8*k)/(ME*pi))*1E2;
        BQ := 2*pi*sqr(a0)*AQ*sqr(Ry)*1E4;

end;


PROCEDURE PRESSURE;

{This procedure calculates the strontium vapour pressure based on a piece-wise
linear approximation to the characteristic vapour pressure curve.
}

begin
    PH := p_ext*(f_He/100);
    PN := p_ext*(f_Ne/100);

    if (Tw >=  1150 ) then
       PS := 0.07
    else if (Tw >=  1100 ) and (Tw < 1150  ) then
       PS := 0.06
    else if (Tw >= 1050  ) and (Tw <  1100 ) then
       PS := 0.05
    else if (Tw >=  1000 ) and (Tw <  1050 ) then
       PS := 0.04
    else if (Tw >=  950 ) and (Tw < 1000  ) then
       PS := 0.03
    else if (Tw >= 900  ) and (Tw < 950  ) then
       PS := 0.02
    else if (Tw < 900 ) then
       PS := 0.01;

end;


PROCEDURE SIMULATE;

{
This is the main simulation procedure
}

var     index, segment, RX, RY                  :integer;
        amp_start, amp_end, sig_centre,
        totseg                                  :integer;
        loss, goseg_on                          :real;
```

```
PROCEDURE PREPARE_RUN;

{
This procedure does the principal initialisation, setting the limits on the
arrays and other variables in the form required by the simulation routine.
It sets the initial axial (time = 0) conditions in a form suitable for use
by SRRAD0.
}

var     index, location, num                    :integer;
        Q                                       :real;

begin           {Start of code section for PREPARE_RUN}

{Initial (time = 0) electric field within plasma}
      E_plasma := 0;
      V_plasma := 0;

{Initial (time = 0) species densities}
      NE := ne_00;
      TE := Tw;
      TG := Tw;
      H1 := PH*1E-4/(k*TG);
      H2 := 0.0;
      H3 := 0.0;
      H4 := 0.0;
      H5 := 0.0;
      H6 := 0.0;
      H7 := 0.0;
      N1 := PN*1E-4/(k*TG);
      N2 := 0.0;
      N3 := 0.0;
      N4 := 0.0;
      N5 := 0.0;
      N6 := 0.0;
      N7 := 0.0;
      S1 := PS*1E-4/(k*TG);
      S2 := ne_00;
      S3 := 0.0;
      S4 := 0.0;
      S5 := 0.0;
      S6 := 0.0;
      S7 := 0.0;
      S8 := 0.0;
      S9 := 0.0;
      S0 := 0.0;
      Np := 0.0;
      I := 0.0;
      P430 := 0.0;
      P416 := 0.0;
      P407 := 0.0;
      P421 := 0.0;
      P1032 := 0.0;
```

```
R1 := 0.0;
R2 := 0.0;
R3 := 0.0;
R4 := 0.0;
R5 := 0.0;
R6 := 0.0;
R7 := 0.0;
R157 := 0.0;
R8 := 0.0;
R9 := 0.0;
R10 := 0.0;
R11 := 0.0;
R12 := 0.0;
R13 := 0.0;
R14 := 0.0;
R15 := 0.0;
R16 := 0.0;
R17 := 0.0;
R18 := 0.0;
R19 := 0.0;
R20 := 0.0;
R151 := 0.0;
R21 := 0.0;
R22 := 0.0;
R23 := 0.0;
R24 := 0.0;
R25 := 0.0;
R26 := 0.0;
R27 := 0.0;
R28 := 0.0;
R29 := 1.30E8;
R30 := 7.1E7;
R31 := 1.46E8;
R32 := 1.20E8;
R33 := 6.9E6;
R34 := 0.0;
R35 := 0.0;
R36 := 0.0;
R37 := 0.0;
R38 := 0.0;
R39 := 0.0;
R40 := 0.0;
R41 := 0.0;
R42 := 0.0;
R43 := 0.0;
R44 := 0.0;
R45 := 0.0;
R46 := 0.0;
R47 := 0.0;
R48 := 0.0;
R49 := 0.0;
R50 := 0.0;
R51 := 0.0;
R52 := 0.0;
R53 := 0.0;
```

```
R54 := 0.0;
R55 := 0.0;
R56 := 0.0;
R57 := 0.0;
R58 := 0.0;
R59 := 0.0;
R60 := 0.0;
R61 := 0.0;
R62 := 0.0;
R63 := 0.0;
R64 := 0.0;
R65 := 0.0;
R66 := 0.0;
R67 := 0.0;
R68 := 0.0;
R69 := 0.0;
R70 := 0.0;
R71 := 0.0;
R72 := 0.0;
R73 := 0.0;
R74 := 0.0;
R75 := 0.0;
R76 := 0.0;
R77 := 0.0;
R78 := 0.0;
R79 := 0.0;
R80 := 0.0;
R81 := 0.0;
R82 := 0.0;
R83 := 0.0;
R84 := 0.0;
R85 := 0.0;
R86 := 0.0;
R87 := 0.0;
R88 := 0.0;
R89 := 0.0;
R90 := 0.0;
R91 := 0.0;
R92 := 0.0;
R93 := 0.0;
R94 := 0.0;
R95 := 0.0;
R96 := 0.0;
R97 := 0.0;
R98 := 0.0;
R99 := 0.0;
R100 := 0.0;
R101 := 0.0;
R102 := 5.24E7;
R103 := 7.5E7;
R104 := 0.0;
R105 := 0.0;
R106 := 0.0;
R107 := 0.0;
R108 := 0.0;
```

```
R109 := 0.0;
R110 := 0.0;
R111 := 0.0;
R112 := 0.0;
R113 := 0.0;
R114 := 0.0;
R115 := 0.0;
R116 := 0.0;
R117 := 0.0;
R118 := 0.0;
R119 := 0.0;
R120 := 0.0;
R121 := 0.0;
R122 := 0.0;
R123 := 0.0;
R124 := 0.0;
R125 := 0.0;
R126 := 0.0;
R127 := 0.0;
R128 := 0.0;
R129 := 0.0;
R130 := 0.0;
R131 := 0.0;
R132 := 0.0;
R133 := 0.0;
R134 := 0.0;
R135 := 0.0;
R136 := 0.0;

{   g0 := 2.64E-16*S5;
    absn := (g0/18) + ((cav_length/amp_length)*2.96E-4*PS);   }

{Initial (time = 0) first time derivatives}

    DNET := 0.0;
    DS1T := 0.0;
    DS2T := 0.0;
    DS3T := 0.0;
    DS4T := 0.0;
    DS5T := 0.0;
    DS6T := 0.0;
    DS7T := 0.0;
    DS8T := 0.0;
    DS9T := 0.0;
    DS0T := 0.0;
    DH1T := 0.0;
    DH2T := 0.0;
    DH3T := 0.0;
    DH4T := 0.0;
    DH5T := 0.0;
    DH6T := 0.0;
    DH7T := 0.0;
    DN1T := 0.0;
    DN2T := 0.0;
    DN3T := 0.0;
```

```
DN4T := 0.0;
DN5T := 0.0;
DN6T := 0.0;
DN7T := 0.0;
DTET := 0.0;
DTGT := 0.0;
DPHT := 0.0;
DPNT := 0.0;
DPST := 0.0;
DNpT := 0.0;
DIT := 0.0;
DP430T := 0.0;
DP416T := 0.0;
DP407T := 0.0;
DP421T := 0.0;
DP1032T := 0.0;

{Initial (time = 0) second time derivatives}

DNES := 0.0;
DS1S := 0.0;
DS2S := 0.0;
DS3S := 0.0;
DS4S := 0.0;
DS5S := 0.0;
DS6S := 0.0;
DS7S := 0.0;
DS8S := 0.0;
DS9S := 0.0;
DS0S := 0.0;
DH1S := 0.0;
DH2S := 0.0;
DH3S := 0.0;
DH4S := 0.0;
DH5S := 0.0;
DH6S := 0.0;
DH7S := 0.0;
DN1S := 0.0;
DN2S := 0.0;
DN3S := 0.0;
DN4S := 0.0;
DN5S := 0.0;
DN6S := 0.0;
DN7S := 0.0;
DTES := 0.0;
DTGS := 0.0;
DPHS := 0.0;
DPNS := 0.0;
DPSS := 0.0;
DNpS := 0.0;
DIS := 0.0;
DP416S := 0.0;
DP430S := 0.0;
DP407S := 0.0;
DP421S := 0.0;
```

```
    DP1032S := 0.0;

{Initial (time = 0) particle velocities}

    VE := SQRT((8*k*TE)/(pi*ME));
    VS := SQRT((8*k*TG)/(pi*MS));
    VH := SQRT((8*k*TG)/(pi*MH));
    VN := SQRT((8*k*TG)/(pi*MN));

{Initial (time = 0) plasma conductivity}

    SP := (NE*sqr(e)*1E6)/((ME*2.30E9*PH*0.76) + (ME*4.80E8*PN*0.76));

{Initial (time = 0) plasma resistivity}

    R_plasma := (amp_length*100)/(AR*SP);

{Initial (time = 0) plasma inductance}

    L_plasma := (amp_length*ME*1E-4)/(NE*sqr(e)*AR);

end;            {End of code section for PREPARE_RUN}


PROCEDURE POTENTIAL;

{
This procedure is called from SIMULATE each time-step, and calculates the
electric potential at time t.
This is then used to calculate the electric field value at time t.
}

begin           {Start of code section for POTENTIAL}

{       if (time <= 250E-9) then
           V_plasma := volt_supply*1000*(time/250E-9)
        else if (time > 250E-9) and (time <= 500E-9) then
           V_plasma := volt_supply*1000*(1-((time-250E-9)/250E-9))
        else if (time > 500E-9) then
           V_plasma := 0;           }                   {Volts}

      E_plasma := (V_plasma*100)/(amp_length);      {Volts/metre}

end;        {Of code section of POTENTIAL}


begin           {Start of code section for SIMULATE}

        time := 0;
        prepare_run;
        header;                 {Printout and display values in use}
        while (time <= (sim_time*1E-9)) do
        begin
           if (round(time*1E12) mod 20000 = 0) then
           begin
```

```
writeln(lst,'                ');
writeln(lst,'time = ',round(time*1E9):11,' nsec:'{,sp2,'Sr2+ = ',S5:11,'cm-3'});
{writeln(lst,'AQ = ',AQ:11,sp2,'BQ = ',BQ:11);                }
writeln(lst,'I = ',I:11,sp2,'DIT = ',DIT:11, 'DST = ',DST:11);
{writeln(lst,'PS = ',PS:11,sp2,'PH = ',PH:11});
writeln(lst,'NE = ',NE:11,sp2,'TE = ',TE:11,sp2,'DTET = ',DTET:11);
writeln(lst,'S1 = ',S1:11,sp2,'S2 = ',S2:11,sp2,'S3 = ',S3:11,sp2,'S4 =',S4:11);
writeln(lst,'S5 = ',S5:11,sp2,'S6 = ',S6:11,sp2,'S7 = ',S7:11);
writeln(lst,'S8 = ',S8:11,sp2,'S9 = ',S9:11,sp2,'S0 = ',S0:11);
writeln(lst,'H1 = ',H1:11,sp2,'H2 = ',H2:11,sp2,'H3 = ',H3:11,sp2,'H4 =',H4:11);
writeln(lst,'H5 = ',H5:11,sp2,'H6 = ',H6:11,sp2,'H7 = ',H7:11);
writeln(lst,'N1 = ',N1:11,sp2,'N2 = ',N2:11,sp2,'N3 = ',N3:11,sp2,'N4 =',N4:11);
writeln(lst,'N5 = ',N5:11,sp2,'N6 = ',N6:11,sp2,'N7 = ',N7:11);
writeln(lst,'P430 = ',P430:11,sp2,'P416 = ',P416:11);
writeln(lst,'E-plasma = ',E_plasma:11,sp2,'V_plasma = ',V_plasma:11);
{writeln(lst,'DNET = ',DNET:11,sp2,'DS5T = ',DS5T:11);      }
writeln(lst,'TG = ',TG:11,sp2,'DTGT = ',DTGT:11);
{writeln(lst,'DS4T = ',DS4T:11,sp2,'DS6T = ',DS6T:11,sp2,'DS7T = ',DS7T:11);
}writeln(lst,'DH2T = ',DH2T:11,sp2,'DH3T = ',DH3T:11);
writeln(lst,'DN3T = ',DN3T:11);
{writeln(lst,'R1 = ',R1:11,sp2,'R2 = ',R2:11,sp2,'R3 = ',R3:11);
writeln(lst,'R4 = ',R4:11,sp2,'R5 = ',R5:11);
writeln(lst,'R6 = ',R6:11,sp2,'R7 = ',R7:11);
writeln(lst,'R10 = ',R10:11,sp2,'R11 = ',R11:11,sp2,'R12 = ',R12:11);
writeln(lst,'R13 = ',R13:11,sp2,'R14 = ',R14:11);
writeln(lst,'R15 = ',R15:11,sp2,'R16 = ',R16:11);
writeln(lst,'R19 = ',R19:11,sp2,'R20 = ',R20:11,sp2,'R21 = ',R21:11);
writeln(lst,'R22 = ',R22:11);
writeln(lst,'R23 = ',R23:11,sp2,'R24 = ',R24:11,sp2,'R25 = ',R25:11);      }
writeln(lst{,'R26 = ',R26:11,sp2,'R29 = ',R29:11,sp2,'R30 = ',R30:11});
writeln(lst{,'R35 = ',R35:11,sp2,'R38 = ',R38:11});
writeln(lst,'R42 = ',R42:11,sp2,'R43 = ',R43:11,sp2,'R48 = ',R48:11);
writeln(lst,'R77 = ',R77:11,sp2,'R80 = ',R80:11);
writeln(lst,'R84 = ',R84:11,sp2,'R85 = ',R85:11,sp2,'R90 = ',R90:11);   {
writeln(lst,'R109 = ',R109:11,sp2,'R113 = ',R113:11,sp2,'R151 = ',R151:11);
writeln(lst,'R157 = ',R157:11);                }
writeln(lst,'S_plasma = ',SP:11,sp2,'R_plasma = ',R_plasma:11);
writeln(lst,'L_plasma = ',L_plasma:11);
{writeln(lst,'Ve = ',VE:11,sp2,'VHe = ',VH:11,sp2,'VSr = ',VS:11);
writeln(lst,'(R19*S1*NE) = ',(R19*S1*NE):11);
writeln(lst,'(R21*S2*NE) =',(R21*S2*NE):11);
writeln(lst,'(R23*S3*NE*NE) = ',(R23*S3*NE*NE):11);
writeln(lst,'(R24*S2*NE*NE)= ',(R24*S2*NE*NE):11);
writeln(lst,'(R42*H1*NE)= ',(R42*H1*NE):11);
writeln(lst,'(R43*H3*NE)= ',(R43*H3*NE):11);
writeln(lst,'(R48*H2*NE*NE)=',(R48*H2*NE*NE):11);
writeln(lst,'(R109*H2*S1)=',(R109*H2*S1):11);
writeln(lst,'(2*R113*H3*S1)=',(2*R113*H3*S1):11);              }
writeln(lst,'21 = ',(R21*S2*NE):11,sp2,'109 = ',(R109*H2*S1):11);
writeln(lst,'113 = ',(R113*H3*S1):11,sp2,'151 = ',(R151*S1*NE):11);
writeln(lst,'6 = ',(R6*S4*NE):11,sp2,'7 = ',(R7*S7*NE):11);
writeln(lst,'17 = ',(R17*S0*NE):11);

        end;
        elec_soln;
```

```
            potential;
            boltzmann;
            scan_amp;
            spec_soln;
            opto_soln;

            write ('Time-step ', round(time*1E12/inc_time), cr);
            time := time + (inc_time*1E-12);
        end;
end;    {Of code section of SIMULATE}


{-------------------------------------------------------------------------}

{This is the main program, Version 1, created on October 17th, 1990}

BEGIN
        install;
        while true do       {Endless loop: terminates by EXIT, 4 lines down}
        begin
            initialise;     {Read in starting values}
            run := ask_input (one_way);
            if not run then exit;       {This is the escape from the WHILE loop}
                constant;               {Calculates constants used in the code}
                pressure;               {Strontium vapour pressure}
                simulate;               {Main simulation procedure}
                {display(false, false, false, true);}
                                        {Output of data to printer}
                {wrapup;}               {Saves files to disk if required}
        end;                            {Of WHILE loop}
END.
```

```
{$B+}    {Boolean complete evaluation on}
{$S+}    {Stack checking on}
{$I+}    {I/O checking on}
{$N+}    {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT SRINIT0;          {Version 1    Created on 17th October 1990}

{$R+}    {Compiler range checking for arrays, etc.}
{$F+}    {Far call compiler directive}
{$O+}    {Overlay compiler directive}


{
 SRINIT0 is an overlaid unit of SRRAD0, a third-generation, zero-dimensional
 strontium vapour laser simulation. This is version 1 of the unit which is
 used to initialise the input parameters of the current SVL system.
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Printer;


TYPE  text_index =  'A'..'~';

CONST cr :char = ^m; lf :char = ^j; ff :char = ^l; bel :char = ^g;
      sp1 = ' '; sp2 = '  ';
      sp3 = '   '; sp4 = '    ';
      sp5 = '     '; sp6 = '      ';
      sp7 = '       '; sp8 = '        ';
      sp9 = '         '; sp10 = '          ';
      sp11 = '           '; sp12 = '            ';
      sp13 = '             '; sp14 = '              ';
      sp15 = '               '; sp16 = '                ';
      sp17 = '                 '; sp18 = '                  ';
      sp19 = '                   '; sp20 = '                    ';
      sp21 = '                     '; sp22 = '                      ';
      sp23 = '                       '; sp24 = '                        ';
      sp25 = '                         ';
      sp26 = '                          ';
      sp27 = '                           ';
      sp28 = '                            ';
      sp29 = '                             ';
      sp30 = '                              ';
      sp31 = '                               ';
      sp32 = '                                ';
      sp33 = '                                 ';
```

```
        sp34 = '                                        ';
        sp35 = '                                        ';
        sp36 = '                                        ';

{Declare the dimensions of the arrays}

{       max_sim_time = 10;   }   {Maximum number of simulation points}

{Declare the fundamental constants of the system}

        pi  = 3.14;             {pi}
        vcl = 3E8;              {Velocity of light in m/s}
        k   = 1.38E-23;         {Boltzmann constant}
        e   = 1.602E-19;        {Electronic charge}
        h   = 6.63E-34;         {Planck's constant}

{Declare the general constants}

        ME  = 9.109E-31;        {Electronic mass}
        MS  = 1.45E-25;         {Strontium atomic mass}
        MH  = 6.68E-27;         {Helium atomic mass}
        MN  = 15.1E-27;         {Neon atomic mass}
        a0  = 0.529E-10;        {Bohr radius}

VAR     line        :       array [text_index] of string [55];

        {ANALYS}            edit_rev, save_data             :char;
                            graph_table                     :string[5];
                            upper_A, upper_B,
                            lower_A, lower_B                :integer;
                            sim_time, inc_time, rad_res,
                            upper_range_H, upper_range_L,
                            lower_range_H, lower_range_L    :real;

        {CONFIG}            amp_width, amp_depth, amp_length,
                            cav_length, mir_ref             :real;

        {ATOMIC}            ne_00                           :real;

        {THERMO}            p_ext, Tw,
                            f_He, f_Sr, f_Ne                :real;


        {ELECTR}            volt_supply,
                            cap_stor, cap_peak,
                            ind_charge, ind_unsat,
                            rel_perm, sat_curr,
                            res_bypass, ind_bypass,
                            res_loop, ind_loop              :real;

        {LASTHY}            res_laser, ind_laser,
                            res_thyr_off, res_thyr_on,
                            ind_thyr, thyr_on_time          :real;
```

```
        title                                                   :string[40];

        run, iterate, one_way, firstime                         :boolean;

{Species array dimensions}

S1, S2, S3, S4, S5,
S6, S7, S8, S9, S0,
H1, H2, H3, H4, H5, H6, H7,
N1, N2, N3, N4, N5, N6, N7,
TG, PH, PS, PN, Np, I,
P430, P416, P407, P421, P1032,
NE, TE,

PEB, AR,
V_plasma, E_plasma,
CL1, CL2,

pump_vol,
g0, absn                                                        :real;

{First time derivative array dimensions}

DS1T, DS2T, DS3T, DS4T, DS5T,
DS6T, DS7T, DS8T, DS9T, DS0T,
DH1T, DH2T, DH3T, DH4T, DH5T, DH6T, DH7T,
DN1T, DN2T, DN3T, DN4T, DN5T, DN6T, DN7T,
DNET,
DP430T, DP416T, DP407T, DP421T, DP1032T,
DTET, DTGT, DPHT, DPNT, DPST, DNpT,
DIT, DST,

{Second time derivative array dimensions}

DS1S, DS2S, DS3S, DS4S, DS5S,
DS6S, DS7S, DS8S, DS9S, DS0S,
DH1S, DH2S, DH3S, DH4S, DH5S, DH6S, DH7S,
DN1S, DN2S, DN3S, DN4S, DN5S, DN6S, DN7S,
DNES,
DTES, DTGS, DPHS, DPNS, DPSS,
DNpS, DIS, DP416S, DP430S, DP407S, DP421S, DP1032S,

{Particle velocity array dimensions}

VE, VS, VH, VN,

{Total plasma conductivity and current density array dimensions}

SP,

{Rate constant array dimensions}

R1, R2, R3, R4, R5,
R6, R7, R8, R9, R10,
R11, R12, R13, R14, R15,
```

```
R16, R17, R18, R19, R20,
R21, R22, R23, R24, R25,
R26, R27, R28, R29, R30,
R31, R32, R33, R34, R35,
R36, R37, R38, R39, R40,
R41, R42, R43, R44, R45,
R46, R47, R48, R49, R50,
R51, R52, R53, R54, R55,
R56, R57, R58, R59, R60,
R61, R62, R63, R64, R65,
R66, R67, R68, R69, R70,
R71, R72, R73, R74, R75,
R76, R77, R78, R79, R80,
R81, R82, R83, R84, R85,
R86, R87, R88, R89, R90,
R91, R92, R93, R94, R95,
R96, R97, R98, R99, R100,
R101, R102, R103, R104, R105,
R106, R107, R108, R109, R110,
R111, R112, R113, R114, R115,
R116, R117, R118, R119, R120,
R121, R122, R123, R124, R125,
R126, R127, R128, R129, R130,
R131, R132, R133, R134, R135,
R136,
R151, R157,

{Miscellaneous array dimensions}

AQ, BQ, Ry                                                    :real;

{Total plasma current, voltage drop, resistance, inductance, capacitance and
conductivity array dimensions}

R_plasma, L_plasma                                            :real;

{State variable solution array dimensions}


{Above are the global arrays to hold amplifier, pulse train & ASE data}

        N                                                     :integer;

        time                                                  :real;

procedure initialise;
function ask_input(var one_way :boolean):boolean;

Implementation

{------------------------------------------------------------------------}

{Now declare the procedures and functions}
```

```
PROCEDURE INITIALISE;

begin     {First set up the array of text strings}

{ ANALYSIS LIMITS }
line ['A'] := 'EDIT/REVIEW PARAMETERS (Y/N):';
line ['B'] := 'MAXIMUM SIMULATION TIME (ns):';
line ['C'] := 'TIME INCREMENT (ps):';
line ['D'] := 'UPPER TRACE A:';
line ['E'] := 'UPPER TRACE B:';
line ['F'] := 'UPPER TRACE RANGE (HIGH/LOW):';
line ['G'] := 'LOWER TRACE A:';
line ['H'] := 'LOWER TRACE B:';
line ['I'] := 'LOWER TRACE RANGE (HIGH/LOW):';
line ['J'] := 'CRT PLOT (GRAPH) OR TABLE (TABLE):';
line ['K'] := 'SAVE DATA TO USER FILE (Y/N):';
line ['L'] := 'RADIAL RESOLUTION REQUIRED (10/20/50 OFF-AXIS POINTS):';


{ (A) CONFIGURATION OF LASER HEAD (CONFIG) }
line ['Q'] := 'DISCHARGE TUBE WIDTH (cm): ';
line ['R'] := 'DISCHARGE TUBE DEPTH (cm): ';
line ['S'] := 'DISCHARGE TUBE LENGTH (ELECTRODE SEPARATION) (cm): ';
line ['T'] := 'LENGTH OF OPTICAL CAVITY (MIRROR SEPARATION) (cm): ';
line ['U'] := 'OUTPUT MIRROR REFLECTIVITY (fraction): ';

{ (B) ATOMIC STATE VARIABLES (ATOMIC) }
line ['W'] := 'INITIAL AXIAL ELECTRON DENSITY (cm-3):';

{ (C) THERMODYNAMIC STATE VARIABLES (THERMO) }
line ['a'] := 'TOTAL EXTERNAL GAS PRESSURE (mbar): ';
line ['b'] := 'DISCHARGE TUBE INNER WALL TEMPERATURE (°K): ';
line ['c'] := 'PERCENTAGE OF HELIUM IN GAS MIX (%): ';
line ['d'] := '                                    ';
line ['e'] := 'PERCENTAGE OF NEON IN GAS MIX (%): ';

{ (D) ELECTRICAL STATE VARIABLES (ELECTR) }
line ['g'] := 'POWER SUPPLY VOLTAGE (kV):';
line ['h'] := 'STORAGE CAPACITANCE (nF): ';
line ['i'] := 'PEAKING CAPACITANCE (nF):';
line ['j'] := 'CHARGING INDUCTANCE (mH):';
line ['k'] := 'UNSATURATED MPC INDUCTANCE (microhenries):';
line ['l'] := 'RELATIVE PERMEABILITY OF FERRITE:';
line ['m'] := 'SATURATION CURRENT (amps):';
line ['n'] := 'BYPASS ELEMENT RESISTANCE (ohms): ';
line ['o'] := 'BYPASS ELEMENT INDUCTANCE (microhenries): ';
line ['p'] := 'DISCHARGE LOOP RESISTANCE (ohms): ';
line ['q'] := 'DISCHARGE LOOP INDUCTANCE (nanohenries): ';

{ (E) LASER AND THYRATRON STATE VARIABLES (LASTHY) }
line ['t'] := 'LASER HEAD RESISTANCE (ohms):';
line ['u'] := 'LASER HEAD INDUCTANCE (nH):';
line ['v'] := 'THYRATRON OFF-RESISTANCE (ohms): ';
line ['w'] := 'ARC-DROP THYRATRON RESISTANCE (ohms): ';
```

```
line ['x'] := 'THYRATRON INDUCTANCE (nH): ';
line ['y'] := 'THYRATRON TURN-ON TIME (ns): ';

    {Now initialise the main parameters to default values}

        edit_rev := 'N';
        sim_time := 3010.0;
        inc_time := 20.0;
        upper_A := 1;
        upper_B := 2;
        upper_range_H := 0.0;
        upper_range_L := 0.0;
        lower_A := 3;
        lower_B := 4;
        lower_range_H := 0.0;
        lower_range_L := 0.0;
        graph_table := 'TABLE';
        save_data := 'N';
        rad_res := 10;

        amp_width := 1.3;
        amp_depth := 1.3;
        amp_length := 50.0;
        cav_length := 150.0;
        mir_ref := 0.3;

        ne_00 := 4E12;

        p_ext := 200.0;
        Tw := 900.0;
        f_He := 100.0;
{       f_Sr := 0.0;    }
        f_Ne := 0.0;

        volt_supply := 7.2;
        cap_stor := 4.0;
        cap_peak := 3.0;
        ind_charge := 150.0;
        res_laser := 1.0;
        ind_laser := 500.0;
        res_thyr_off := 100000.0;
        res_thyr_on := 2.0;
        ind_thyr := 500.0;
        thyr_on_time := 5.0;
        res_bypass := 5.0;
        ind_bypass := 100.0;
        res_loop := 1.0;
        ind_loop := 1.0;
        ind_unsat := 100.0;
        sat_curr := 75.0;
        rel_perm := 500;

    firstime := false;
    one_way := false;
end;
```

```
FUNCTION  ask_input (var one_way :boolean):boolean;

{
 Input of data to SRRAD0.  Contains a nested procedure param designed to act
 like a screen editor.
}

var       index                                          :text_index;
          key                                            :char;


procedure param (which :text_index; edit :boolean);

{
Parameter input procedure designed to behave like a screen editor.  The
calling parameter WHICH specifies which of the model parameters is involved,
and EDIT specifies whether it is to be displayed or edited.  Each parameter
is input at a distinct place on the screen, and the cursor is moved there to
allow the new value to be entered after its prompt.  The prompt doubles as a
descriptor of the quantity for display purposes.
}

var       row       :integer;

begin     {Start of code section for PARAM.  First work out where to put the
           prompt on the screen}

     if which in ['A'..'L']
        then row := (ORD (which) - ORD ('A'))
     else if which in ['Q'..'U']
        then row := (ORD (which) - ORD ('Q'))
     else if which = 'W'
        then row := (ORD (which) - ORD ('W'))
     else if which in ['a'..'e']
        then row := (ORD (which) - ORD ('a'))
     else if which in ['f'..'s']
        then row := (ORD (which) - ORD ('f'))
     else if which in ['t'..'{']
        then row := (ORD (which) - ORD ('t'));
     row := row + 6;
     gotoxy (1,row);
     clreol;                {Erase previous value of the prompt and value}
        if (which in ['A'..'L']) or (which in ['Q'..'U']) or
          (which = 'W') or (which in ['a'..'e']) or
          (which in ['f'..'s']) or (which in ['t'..'{'])
        then write (sp10, which + '....' + line [which]  );   {Put up new prompt}

     case which of
          'A'  : if edit
                    then read (edit_rev)
                    else write (edit_rev:30);
          'B'  : if edit
                    then read (sim_time)
                    else write (sim_time:30:2);
```

```
'C'  : if edit
         then read (inc_time)
         else write (inc_time:39:2);
'D'  : if edit
         then read (upper_A)
         else write (upper_A:45);
'E'  : if edit
         then read (upper_B)
         else write (upper_B:45);
'F'  : if edit
         then read (upper_range_H, upper_range_L)
         else write (upper_range_H:15:2, upper_range_L:15:2);
'G'  : if edit
         then read (lower_A)
         else write (lower_A:45);
'H'  : if edit
         then read (lower_B)
         else write (lower_B:45);
'I'  : if edit
         then read (lower_range_H, lower_range_L)
         else write (lower_range_H:15:2, lower_range_L:15:2);
'J'  : if edit
         then read (graph_table)
         else write (graph_table:25);
'K'  : if edit
         then read (save_data)
         else write (save_data:30);
'L'  : if edit
         then read (rad_res)
         else write (rad_res:5:0);
'Q'  : if edit
         then read (amp_width)
         else write (sp26,amp_width:6:2);
'R'  : if edit
         then read (amp_depth)
         else write (sp26,amp_depth:6:2);
'S'  : if edit
         then read (amp_length)
         else write (sp2, amp_length:6:2);
'T'  : if edit
         then read (cav_length)
         else write (sp2, cav_length:6:2);
'U'  : if edit
         then read (mir_ref)
         else write (sp14, mir_ref:6:2);
'W'  : if edit
         then read (ne_00)
         else write (sp10, ne_00:11);
'a'  : if edit
         then read (p_ext)
         else write (sp16, p_ext:6:2);
'b'  : if edit
         then read (Tw)
         else write (sp1, Tw:6:2);
'c'  : if edit
```

```
            then read (f_He)
            else write (sp15, f_He:6:2);
'd'   : if edit
            then read (f_Sr)
            else write (sp12, f_Sr:6:2);
'e'   : if edit
            then read (f_Ne)
            else write (sp17, f_Ne:6:2);
'g'   : if edit
            then read (volt_supply)
            else write (sp27, volt_supply:6:2);
'h'   : if edit
            then read (cap_stor)
            else write (sp27, cap_stor:6:2);
'i'   : if edit
            then read (cap_peak)
            else write (sp28, cap_peak:6:2);
'j'   : if edit
            then read (ind_charge)
            else write (sp28, ind_charge:6:2);
'k'   : if edit
            then read (ind_unsat)
            else write (sp11, ind_unsat:6:2);
'l'   : if edit
            then read (rel_perm)
            else write (sp19, rel_perm:6:2);
'm'   : if edit
            then read (sat_curr)
            else write (sp27, sat_curr:6:2);
'n'   : if edit
            then read (res_bypass)
            else write (sp20, res_bypass:5:1);
'o'   : if edit
            then read (ind_bypass)
            else write (sp12, ind_bypass:5:1);
'p'   : if edit
            then read (res_loop)
            else write (sp20, res_loop:5:1);
'q'   : if edit
            then read (ind_loop)
            else write (sp14, ind_loop:4:1);
't'   : if edit
            then read (res_laser)
            else write (sp16,res_laser:6:2);
'u'   : if edit
            then read (ind_laser)
            else write (sp26,ind_laser:6:2);
'v'   : if edit
            then read (res_thyr_off)
            else write (sp17,res_thyr_off:6:2);
'w'   : if edit
            then read (res_thyr_on)
            else write (sp15,res_thyr_on:6:2);
'x'   : if edit
            then read (ind_thyr)
```

```
                else write (sp26,ind_thyr:6:2);
        'y'   : if edit
                then read (thyr_on_time)
                else write (sp24,thyr_on_time:6:2);

    end;  {Case statement on which}

end;      {End of code section for PARAM}


procedure edit_prompt1;

{
Nested procedure called from ASK_INPUT relating to the editing of analysis
limits
}

begin     {Start of code section for EDIT_PROMPT1}

    gotoxy (1,24);
    clreol;
    write ('Hit menu to edit a value, 1: edit a parameter, 2: run, 3: quit');

end;      {End of code section for EDIT_PROMPT1}


procedure edit_prompt2;

{
Nested procedure called from ASK_INPUT relating to the editing of operating
parameters
}

begin     {Start of code section for EDIT_PROMPT2}

    gotoxy (1,24);
    clreol;
    write ('Hit menu to edit a value, 1: view next page, 2: run, 3: quit');

end;      {End of code section for EDIT_PROMPT2}


procedure laserthyr;

var   index                                   :text_index;

begin
        clrscr;
        gotoxy (1,3);
        writeln (sp4,title,': LASER AND THYRATRON PARAMETERS',cr,lf);
        for index := 't' to 'y' do
            param (index, firstime);
        repeat
            for index := 't' to 'y' do
                param (index, false);
```

```
                gotoxy (1,24);
                clreol;
                write ('Hit menu to edit a value, 2: run, or 3: quit');
                Key := Readkey;
{! 5. USE TURBO3 ^unit for access to KBD, or instead use CRT and ReadKey}
                if key in ['t'..'y']
                     then param (key, true);
                if key in ['3']
                     then halt;
                if key in ['2']
                     then ask_input := true;
           until key in ['2'];

           gotoxy (1,24);
           write ('Do you want to iterate with increasing input parameters? Y/N');

end;    {Of ASK_INPUT function}

procedure electr;

var     index                                   :text_index;

begin
           clrscr;
           gotoxy (1,3);
           writeln (sp4, title, ': ELECTRICAL PARAMETERS', cr, lf);
           for index := 'g' to 'q' do
                param (index, firstime);
           repeat
                for index := 'g' to 'q' do
                     param (index, false);
                edit_prompt2;
                Key := Readkey;
{! 4. USE TURBO3 ^unit for access to KBD, or instead use CRT and ReadKey}
                if key in ['g'..'q']
                   then param (key, true);
                if key in ['3']
                   then halt;
                if key in ['2']
                   then ask_input := true;
                if key in ['1'] then
                   begin
                     laserthyr;
                   end;
           until key in ['2'];
end;

procedure thermo;

var     index                                   :text_index;

begin
           clrscr;
           gotoxy (1,3);
           writeln (sp4, title, ': THERMODYNAMIC PARAMETERS', cr, lf);
```

```
          for index := 'a' to 'e' do
               param (index, firstime);
          repeat
               for index := 'a' to 'e' do
                    param (index, false);
               edit_prompt2;
               Key := Readkey;
{! 4. USE TURBO3 ^unit for access to KBD, or instead use CRT and ReadKey}
               if key in ['a'..'e']
                    then param (key, true);
               if key in ['3']
                    then halt;
               if key in ['2']
                    then ask_input := true;
               if key in ['1'] then
                    begin
                       electr;
                    end;
          until key in ['2'];
end;

procedure atomic;

var       index                                   :text_index;

begin
          clrscr;
          gotoxy (1,3);
          writeln (sp6, title ,': ATOMIC PARAMETERS', cr, lf);
          index := 'W';
               param (index, firstime);
          repeat
               index := 'W';
                  param (index, false);
               edit_prompt2;
               Key := Readkey;
{! 3. USE TURBO3 ^unit for access to KBD, or instead use CRT and ReadKey}
               if key in ['W']
                  then param (key, true);
               if key in ['3']
                  then halt;
               if key in ['2']
                  then ask_input := true;
               if key in ['1'] then
                  begin
                    thermo;
                  end;
          until key in ['2'];
end;

procedure config;

var       index                                   :text_index;

begin
```

```
            clrscr;
            gotoxy (1,3);
            writeln (sp4, title ,': CONFIGURATION OF LASER HEAD', cr, lf);
            for index := 'Q' to 'T' do
                 param (index, firstime);
            repeat
                 for index := 'Q' to 'U' do
                    param (index, false);
                 edit_prompt2;
                 Key := Readkey;
{! 2. USE TURBO3 ^unit for access to KBD, or instead use CRT and ReadKey}
                 if key in ['Q'..'U']
                    then param (key, true);
                 if key in ['3']
                    then halt;
                 if key in ['2']
                    then ask_input := true;
                 if key in ['1'] then
                    begin
                      atomic;
                    end;
            until key in ['2'];
end;


begin     {Start of code section for ASK_INPUT}

     ask_input := false;
     clrscr;
  writeln('=================================================',lf);
  writeln('KINETIC MODELLING OF He-Sr RECOMBINATION LASERS',lf);
  writeln('=================================================',lf);
  writeln;
  writeln('SRRAD is a two-dimensional SVL simulation');
  writeln('program designed to investigate the kinetic processes');
  writeln('occuring within the active medium',lf);
  writeln('Title of simulation: ');
  readln (title);

     clrscr;
     gotoxy (1,3);
     writeln (sp10, title ,': ANALYSIS LIMITS', cr, lf);
     for index := 'A' to 'L' do
          param (index, firstime);
     repeat
          for index := 'A' to 'L' do
          param (index, false);
          edit_prompt1;
          Key := Readkey;
{! 1. USE TURBO3 ^unit for access to KBD, or instead use CRT and ReadKey}
          if key in ['A'..'L']
               then param (key, true);
          if key in ['3']
               then halt;
          if key in ['2']
```

```
            then ask_input := true;
        if key in ['1'] then config;

    until key in ['2'];

        firstime := false;     {So next time round we show the values at once}

end;            {Of ASK_INPUT function}

END.
```

```
{$B+}    {Boolean complete evaluation on}
{$S+}    {Stack checking on}
{$I+}    {I/O checking on}
{$N+}    {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT SRHEAD0;        {Version 1    Created on 17th October 1990}

{$R+}    {Compiler range checking for arrays, etc.}
{$F+}    {Far call compiler directive}
{$O+}    {Overlay compiler directive}


{
 SRHEADER is an overlaid unit of SRRAD0, a zero-dimensional SVL kinetics
 simulation.  This is version 1 of the unit which is used to provide
 both an on-screen and hard copy printout of the parameters in use in the
 current simulation.  An additional line on the screen keeps the user
 informed of what's happening when a run is in progress.  SRHEADER contains
 a nested procedure, SCREEN.
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Printer,
  SrInit0;


TYPE     textline = string [80];

VAR      number   :          string [10];

procedure header;

Implementation

{-------------------------------------------------------------------------}

{Now declare the procedures and functions}

PROCEDURE HEADER;                    {Printout and display values in use}

procedure SCREEN (outstring :textline);     {Writes to both screen and printer}

begin          {Start of code section for SCREEN procedure}
    writeln(outstring, lf);
    writeln(lst, outstring, lf);
end;           {End of code section for SCREEN}
```

```
begin           {Start of code section for HEADER procedure}
    clrscr;
    writeln(lst,'==========================================');
    writeln(lst,title);
    writeln(lst,'==========================================',lf);
    writeln(lst);

    str(amp_width:3:1, number);
    screen(line['Q'] + number);
    str(amp_depth:3:1, number);
    screen(line['R'] + number);
    str(amp_length:3:1, number);
    screen(line['S'] + number);
    writeln(lst,'__________________________________________',lf);
    writeln(lst);

    str(ne_00:10, number);
    screen(line['W'] + number);
    str(p_ext:3:1, number);
    screen(line['a'] + number);
    str(f_He:3:1, number);
    screen(line['c'] + number);
    str(f_Ne:3:1, number);
    screen(line['e'] + number);
    writeln(lst,'__________________________________________',lf);
    writeln(lst);

    str(volt_supply:10, number);
    screen(line['g'] + number);
    str(cap_stor:3:1, number);
    screen(line['h'] + number);
    writeln(lst,'==========================================',lf);
    writeln(lst);
writeln(lf, 'Run in progress with the above values: ');
writeln(lst,                round(sim_time*1000/inc_time),' steps',lf);

end;            {Of HEADER procedure}

END.
```

```
{$B+}    {Boolean complete evaluation on}
{$S+}    {Stack checking on}
{$I+}    {I/O checking on}
{$N+}    {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT SRELEC0;          {Version 1    Created on 17th October 1990}

{$R+}    {Compiler range checking for arrays, etc.}
{$F+}    {Far call compiler directive}
{$O+}    {Overlay compiler directive}

{
 SRELEC0 is an overlaid unit of SRRAD0, a third-generation, zero-dimensional
 strontium vapour laser simulation.  This is version 1 of the unit which
 accepts the plasma resistance and inductance from the main program and
 solves the electrical equations describing the discharge circuitry.
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Printer,
  Srinit0;


CONST

{Declare the dimensions of the arrays}

        max_A_dim = 5;    {Maximum dimensions of A and B arrays}
        max_B_dim = 1;    {Maximum dimension of B array}


VAR     {ELECTR}
        ind_sat,
        R_thyr, LS                          :real;

{Solution array dimensions}

RL, LL,
RT, LB, LT, RB, CS,
U, V_thyr,
VCS, ILL, ILB,
DVCST, DILLT, DILBT,
T1, T2, T3, T4, T5,
T6, T7, T8, T9                              :real;

AC1, AC2, AC3, AC4, AC5                     :real;
```

```
{Total plasma current, voltage drop, resistance, inductance, capacitance and
conductivity array dimensions}

procedure elec_soln;
procedure finite_diff;

Implementation

{Above are the global arrays to hold amplifier, pulse train & ASE data}

{-------------------------------------------------------------------------}

{Now declare the procedures and functions}


procedure finite_diff;

{This procedure is called from ELEC_SOLN each time-step, and is solves the
electrical equations by means of finite differences.
}

begin           {Start of code section for FINITE_DIFF}

    DVCST := (T1*VCS) + (T2*ILL) + (T3*ILB);
    DILLT := (T4*VCS) + (T5*ILL) + (T6*ILB);
    DILBT := (T7*VCS) + (T8*ILL) + (T9*ILB);

    VCS := VCS + (inc_time*1E-12*DVCST);
    ILL := ILL + (inc_time*1E-12*DILLT);
    ILB := ILB + (inc_time*1E-12*DILBT);
    V_thyr := VCS - V_plasma;

if (round(time*1E12) mod 20000 = 0) then
begin                                   {Output of data to printer}
    writeln(lst,'time = ', (time*1E9):11,' ns');

    writeln (lst,'Voltage across storage capacitor = ',VCS:11,' V');
    writeln (lst,'Voltage across thyratron = ',V_thyr:11,' V');
    writeln (lst,'Voltage across laser head = ',V_plasma:11,' V');

    writeln (lst,'Current in laser head = ',ILL:11,' A');
    writeln (lst,'Current in thyratron = ',(ILL + ILB):11,' A');
    writeln (lst,'Current in bypass element = ',ILB:11,' A');

    writeln (lst,'Resistance of laser head = ',RL:11,' ohms');
    writeln (lst,'Resistance of thyratron = ',RT:11,' ohms');
    writeln (lst,'Resistance of bypass element = ',RB:11,' ohms');

    writeln (lst,'Inductance of laser head = ',LL:11,' henries');
    writeln (lst,'Inductance of MPC switch = ',LB:11,' henries');
    writeln (lst,'Inductance of thyratron = ',LT:11,' henries');

    writeln (lst,'Capacitance of storage capacitor = ',CS:11,' nanofarads');
```

```
    writeln(lst,'      ');
end;

    {wrapup;}                          {Saves files to disk if required}

end;      {Of code section of FINITE_DIFF}


PROCEDURE ELEC_SOLN;

{
Calculation of electrical state variables
}

var       RX, RY                              :integer;

begin          {Start of code section for ELEC_SOLN}

          if time = 0.0 then
          begin
             VCS := 2*volt_supply*1E3;
             ILL := 0.0;
             ILB := 0.0;
             V_thyr := VCS - V_plasma;

             LB := ind_bypass*1E-6;
             RL := (res_laser + R_plasma);
             LL := ((ind_laser*1E-9) + L_plasma);
             LT := ind_thyr*1E-9;
             RB := res_bypass;
             RT := res_thyr_off;
             CS := cap_stor*1E-9;
             ind_sat := LB/rel_perm;

             AC1 := (LL*LB) + (LT*LB) + (LL*LT);
             AC2 := (RL*LB) + (RT*LB) + (RL*LT);
             AC3 := (RB*LT) - (RT*LB);
             AC4 := (RB*LL) + (RT*LL) + (LT*RB);
             AC5 := (RL*LT) - (RT*LL);

             T1 := 0.0;
             T2 := -1/CS;
             T3 := -1/CS;

             T4 := LB/AC1;
             T5 := -AC2/AC1;
             T6 := AC3/AC1;

             T7 := LL/AC1;
             T8 := AC5/AC1;
             T9 := -AC4/AC1;

{    writeln(lst,'AC1 = ',AC1:11);
     writeln(lst,'AC2 = ',AC2:11);
     writeln(lst,'AC3 = ',AC3:11);
```

```
        writeln(lst,'AC4 = ',AC4:11);
        writeln(lst,'AC5 = ',AC5:11);
        writeln(lst,'    ');

        writeln(lst,'T1 = 0.0');
        writeln(lst,'T2 = ',(-1/CS):11);
        writeln(lst,'T3 = ',(-1/CS):11);
        writeln(lst,'T4 = ',(LB/AC1):11);
        writeln(lst,'T5 = ',(-AC2/AC1):11);
        writeln(lst,'T6 = ',(AC3/AC1):11);
        writeln(lst,'T7 = ',(LL/AC1):11);
        writeln(lst,'T8 = ',(AC5/AC1):11);
        writeln(lst,'T9 = ',(-AC4/AC1):11);
        writeln(lst,'    ');
}
            end;

        U := 0.0;

{           if (sat_curr-ILB) > 5 then
              LB := ind_unsat*1E-6
            else if (((sat_curr-ILB) <= 5) and ((sat_curr-ILB) >= -5)) then
              begin
                LB := (ind_unsat*1E-6*((sat_curr+5-ILB)/(sat_curr+5)))
                LB := LB + (ind_sat*1E-6);
              end;
}
            if ILB < sat_curr then
              LB := ind_unsat*1E-6
            else LB := ind_sat;

            RL := res_laser + R_plasma;
            LL := (ind_laser*1E-9) + L_plasma;

            RT := res_thyr_off * exp(-time/(thyr_on_time*1E-9));
            if (time > (9*thyr_on_time*1E-9)) then
              RT := res_thyr_on;
            if (ILL + ILB) < -0.1 then
              RT := res_thyr_off;

            AC1 := (LL*LB) + (LT*LB) + (LL*LT);
            AC2 := (RL*LB) + (RT*LB) + (RL*LT);
            AC3 := (RB*LT) - (RT*LB);
            AC4 := (RB*LL) + (RT*LL) + (LT*RB);
            AC5 := (RL*LT) - (RT*LL);

            T1 := 0.0;
            T2 := -1/CS;
            T3 := -1/CS;

            T4 := LB/AC1;
            T5 := -AC2/AC1;
            T6 := AC3/AC1;

            T7 := LL/AC1;
```

```
            T8 := AC5/AC1;
            T9 := -AC4/AC1;

            finite_diff;

            V_plasma := LL*(((LB/AC1)*VCS) - ((LB/AC1)*U) - ((AC2/AC1)*ILL)
            V_plasma := V_plasma + ((AC3/AC1)*ILB)) + (ILL*RL);

end;        {Of code section of ELEC_SOLN}

END.
```

```
{$B+}    {Boolean complete evaluation on}
{$S+}    {Stack checking on}
{$I+}    {I/O checking on}
{$N+}    {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT SRTEMP0;          {Version 1     Created on 17th October 1990}

{$R+}    {Compiler range checking for arrays, etc.}
{$F+}    {Far call compiler directive}
{$O+}    {Overlay compiler directive}

{
 SRTEMP0 is an overlaid unit of SRRAD0, a third-generation, zero-dimensional
 strontium vapour laser simulation.  This is version 1 of the unit which
 calculates the electron and heavy particle temperatures assuming a
 Maxwellian electron energy distribution.
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Printer,
  SrInit0,
  SrRate0;


procedure boltzmann;

Implementation

{-------------------------------------------------------------------------}

{Now declare the procedures and functions}


PROCEDURE BOLTZMANN;

{This procedure is called from SIMULATE each time-step, and assumes a
Maxwellian electron energy distribution at each time-step in the
simulation, to obtain the electron/heavy particle temperature at each position}

begin                  {Start of code section for BOLTZMANN}

{First time derivatives - rate equations}
if (time>1E-6) and (round(time*1E12) mod 5000 = 0) then
begin
    writeln(lst,'Te = ',TE:11,sp2,'dTe/dt = ',DTET:11);
    writeln(lst,'Tg = ',TG:11,sp2,'dTg/dt = ',DTGT:11);
```

```
end;
{     if (E_plasma < 0.0) then
         DTET := 0.0
      else        }
DTET:=
2*sqr(E_plasma)*sqr(e)/((3*k*ME*2.30E9*PH*0.76)+(3*k*ME*4.80E8*PN*0.76));
                        {Energy gain due to longitudinal electric-field input}
if (time>1E-6) and (round(time*1E12) mod 50000 = 0) then
    writeln(lst,'dTe/dt (E-field) = ',DTET:11);
DTET := DTET - (2*ME*(TE-TG)*((((H1+H2+H3+H4)/MH)*0.5E-15*VE*100));
DTET := DTET - (2*ME*(TE-TG)*(((N1+N2+N3+N4)/MN)*0.25E-16*VE*100));
                        {Energy loss due to elastic collisions}
if (time>1E-6) and (round(time*1E12) mod 50000 = 0) then
    writeln(lst,'dTe/dt (Elastic collisions) = ',DTET:11);
DTET := DTET + ((2/(3*k))*((R10*S4*(ES4-ES2)*e) + (R11*S7*(ES7-ES2)*e)));
DTET := DTET + ((2/(3*k))*((R12*S4*(ES4-ES6)*e) + (R13*S4*(ES4-ES8)*e)));
DTET := DTET + ((2/(3*k))*((R14*S7*(ES7-ES8)*e) + (R15*S5*(ES5-ES4)*e)));
DTET := DTET + ((2/(3*k))*((R16*S5*(ES5-ES7)*e) + (R23*S3*(ES3-ES0)*e)));
DTET := DTET + ((2/(3*k))*((R24*S2*(ES2-ES9)*e) + (R17*S0*(ES0-ES5)*e)));
DTET := DTET + ((2/(3*k))*((R18*S9*(ES9-ES1)*e) + (R38*H3*(EH3-EH1)*e)));
DTET := DTET + ((2/(3*k))*((R48*H2*NE*(EH2-EH4)*e)));
DTET := DTET + ((2/(3*k))*((R109*S1*(H2/NE)*(EH2-ES3)*e)));
DTET := DTET + ((2/(3*k))*((R39*H4*(EH4-EH1)*e)+(R40*H4*(EH4-EH3)*e)));
DTET := DTET + ((2/(3*k))*((R49*H2*NE*(EH2-EH6)*e)));
DTET := DTET + ((2/(3*k))*((R50*H5*NE*(EH5-EH6)*e)));
DTET := DTET + ((2/(3*k))*((R80*N3*(EN3-EN1)*e) + (R90*H2*(EN2-EN4)*e)));
DTET := DTET + ((2/(3*k))*((R115*S1*(N2/NE)*(EN2-ES3)*e)));
DTET := DTET + ((2/(3*k))*((R81*N4*(EN4-EN1)*e)+(R82*N4*(EN4-EN3)*e)));
                        {Energy gain due to superelastic collisions}
if (time>1E-6) and (round(time*1E12) mod 50000 = 0) then
    writeln(lst,'dTe/dt (Superelastic collisions) = ',DTET:11);
DTET := DTET + ((2/(3*k))*((R113*(H3/NE)*S1*(EH3-ES3)*e)));
DTET := DTET + ((2/(3*k))*((R119*(N3/NE)*S1*(EN3-ES3)*e)));
                        {Energy gain due to Penning ionisations}
if (round(time*1E12) mod 50000 = 0) then
    writeln(lst,'dTe/dt (Penning ionisations) = ',DTET:11);
DTET := DTET - ((2/(3*k))*((R1*S2*(ES4-ES2)*e) + (R2*S2*(ES7-ES2)*e)));
DTET := DTET - ((2/(3*k))*((R3*S6*(ES4-ES6)*e) + (R4*S8*(ES4-ES8)*e)));
DTET := DTET - ((2/(3*k))*((R5*S8*(ES7-ES8)*e) + (R6*S4*(ES5-ES4)*e)));
DTET := DTET - ((2/(3*k))*((R7*S7*(ES5-ES7)*e) + (R19*S1*(ES2-ES1)*e)));
DTET := DTET - ((2/(3*k))*((R21*S2*(ES3-ES2)*e) + (R8*S5*(ES0-ES5)*e)));
DTET := DTET - ((2/(3*k))*((R9*S1*(ES9-ES1)*e) + (R157*S4*(ES0-ES4)*e)));
DTET := DTET - ((2/(3*k))*((R20*S9*(ES2-ES9)*e) + (R22*S0*(ES3-ES0)*e)));
DTET := DTET - ((2/(3*k))*((R151*S1*(ES3-ES1)*e)+(R35*H1*(EH3-EH1)*e)));
DTET := DTET - ((2/(3*k))*((R42*H1*(EH2-EH1)*e)+(R43*H3*(EH2-EH3)*e)));
DTET := DTET - ((2/(3*k))*((R36*H1*(EH4-EH1)*e) + (R37*H3*(EH4-EH3)*e)));
DTET := DTET - ((2/(3*k))*((R45*H2*(EH7-EH2)*e)+(R77*N1*(EN3-EN1)*e)));
DTET := DTET - ((2/(3*k))*((R84*N1*(EN2-EN1)*e)+(R85*N3*(EN2-EN3)*e)));
DTET := DTET - ((2/(3*k))*((R78*N1*(EN4-EN1)*e) + (R79*N3*(EN4-EN3)*e)));
DTET := DTET - ((2/(3*k))*((R87*N2*(EN7-EN2)*e)));
                        {Energy loss due to inelastic collisions}
if (time>1E-6) and (round(time*1E12) mod 50000 = 0) then
    writeln(lst,'dTe/dt (Inlastic collisions) = ',DTET:11);
if PN > 0.0 then
DTGT:=
```

```
2*ME*(TE-TG)*(((H1+H2+H3+H4)/MH)*0.5E-15*VE*100*NE/(H1+H2+H3+H4));
DTGT:=DTGT+
(2*ME*(TE-TG)*(((N1+N2+N3+N4)/MN)*0.25E-16*VE*100*NE/(N1+N2+N3+N4)));
else
DTGT:=
2*(ME/MH)*(TE-TG)*((H1+H2+H3+H4)*0.5E-15*VE*100*NE/(H1+H2+H3+H4));
    DPHT := 0.0;
    DPNT := 0.0;
{   DPST := 0.0;          }

{Second time derivatives}

{   DTES   := (DTET - DTET)/(inc_time*1E-12);
    DTGS   := (DTGT - DTGT)/(inc_time*1E-12);
    DPHS   := (DPHT - DPHT)/(inc_time*1E-12);
    DPNS   := (DPNT - DPNT)/(inc_time*1E-12);
    DPSS   := (DPST - DPST)/(inc_time*1E-12);
}
{Solution for electron and gas temperatures, and krypton, argon and fluorine
pressures}

TE := TE + (DTET*inc_time*1E-12) + (DTES*0.5*SQR(inc_time*1E-12));
TG := TG + (DTGT*inc_time*1E-12) + (DTGS*0.5*SQR(inc_time*1E-12));
PH := PH + (DPHT*inc_time*1E-12) + (DPHS*0.5*SQR(inc_time*1E-12));
PN := PN + (DPNT*inc_time*1E-12) + (DPNS*0.5*SQR(inc_time*1E-12));
PS := PS + (DPST*inc_time*1E-12) + (DPSS*0.5*SQR(inc_time*1E-12));


{Particle velocities}
    VE := SQRT((8*k*TE)/(pi*ME));
    VS := SQRT((8*k*TG)/(pi*MS));
    VH := SQRT((8*k*TG)/(pi*MH));
    VN := SQRT((8*k*TG)/(pi*MN));

end;    {Of code section of BOLTZMANN}

END.
```

```
{$B+}    {Boolean complete evaluation on}
{$S+}    {Stack checking on}
{$I+}    {I/O checking on}
{$N+}    {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT SRRATE0;          {Version 1     Created on 17th October 1990}

{$R+}    {Compiler range checking for arrays, etc.}
{$F+}    {Far call compiler directive}
{$O+}    {Overlay compiler directive}

{
SRRATE0 is an overlaid unit of SRRAD0, a third-generation, zero-dimensional
strontium vapour laser simulation.  This is version 1 of the unit which
calculates the plasma kinetic properties - rate constants and first and
second time derivatives.
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Printer,
  Srinit0;

CONST

{Declare the general constants}

{Declare the cross-sections and oscillator strengths}

      f1   = 0.71;
      f2   = 0.34;
      f3   = 0.096;
      f4   = 0.016;
      f5   = 0.084;
      f6   = 0.180;
      f7   = 0.093;
{     f157 = 1;
      f8   = 1;
      f9   = 1;            }

      e19  = 7.5E-16;
{     e20  = 1;            }
      e151 = 5.0E-16;
      e21  = 2.5E-16;
{     e22  = 1;            }

      e35  = 6.2E-18;
```

```
      e36  = 4.5E-18;
      e37  = 4.5E-18;

      e42  = 0.6E-16;
      e43  = 6.5E-16;
{     e44  = 1;          }
      e45  = 0.05E-16;

      e61  = 1.6E-16;
      e62  = 3.3E-16;
{     e63  = 1;          }

      e77  = 9.1E-18;

      e81  = 1.0E-13;
      e82  = 1.0E-13;

      e84  = 1.8E-16;
      e85  = 6.0E-16;
{     e86  = 1;          }
      e87  = 0.42E-16;

      e98  = 1.4E-18;

      e104 = 2.3E-18;
      e105 = 1.9E-17;

      e109 = 2.0E-15;
{     e110 = 1;
      e111 = 1;
      e112 = 1;          }
      e113 = 2.0E-15;
{     e114 = 1;          }

      e115 = 0.5E-15;
{     e116 = 1;
      e117 = 1;
      e118 = 1;          }
      e119 = 0.5E-15;
{     e120 = 1;          }

      e123 = 1.5E-18;
{     e124 = 1;          }

      e126 = 7.1E-16;
      e127 = 6.0E-16;
      e128 = 1.5E-15;

{Declare the stimulated and spontaneous emission probabilities}

      A29 = 1.30E8;
      A30 = 7.1E7;
      A31 = 1.46E8;
      A32 = 1.20E8;
      A33 = 6.9E6;
```

```
{    A69 = 1;
     A70 = 1;
     A71 = 1;                    }
     A102= 5.4E6;
     A103= 3.0E6;

{Declare the absorption probabilities}

{    B34  = 1;                   }

{Declare the cross-section for ionisation}

     emi  = 1.0E-16;

{Declare the energy level separations (eV)}

     ES1 = 0.0;
     ES2 = 5.9;
     ES3 = 16.5;
     ES4 = 9.0;
     ES5 = 11.0;
     ES6 = 7.0;
     ES7 = 9.1;
     ES8 = 7.1;
     ES9 = 4.5;
     ES0 = 12.6;
     EH1 = 0.0;
     EH2 = 24.5;
     EH3 = 20.0;
     EH4 = 21.1;
     EH5 = 23.5;
     EH6 = 17.5;
     EH7 = 32.5;
     EN1 = 0.0;
     EN2 = 21.6;
     EN3 = 16.6;
     EN4 = 18.5;
     EN5 = 20.5;
     EN6 = 14.5;
     EN7 = 30.0;
     ENE = 0.0;

{Declare the level degeneracies}

     gS1 = 2;
     gS2 = 2;
     gS3 = 2;
     gS4 = 4;
     gS5 = 2;
     gS6 = 6;
     gS7 = 2;
     gS8 = 4;
     gS9 = 1;
     gS0 = 1;
     gH1 = 2;
```

```
      gH2 = 2;
      gH3 = 2;
      gH4 = 1;
      gH5 = 1;
      gH6 = 1;
      gH7 = 2;
      gN1 = 2;
      gN2 = 2;
      gN3 = 2;
      gN4 = 1;
      gN5 = 1;
      gN6 = 1;
      gN7 = 2;

procedure rate_constants;
procedure rate_equations;
procedure time_derivs;
procedure spec_soln;

Implementation

{------------------------------------------------------------------------}

{Now declare the procedures and functions}


PROCEDURE RATE_CONSTANTS;

{This procedure is called from SIMULATE each time-step, and calculates the
rate constants at each time step in the simulation, based on the electron/
particle temperature, energy level separation and transition oscillator
strength at each position}

var  TE_to_0_43, TE_to_0_50, TE_to_0_70,
     TE_to_0_73, TE_to_4_5, TE_to_3_0,
     TG_to_0_167, TG_to_0_50                          :real;

begin                    {Start of code section for RATE_CONSTANTS}

{Calculation of the electron temperature raised to a power}

    TE_to_0_43 := exp(0.43*ln(TE));
    TE_to_0_50 := exp(0.50*ln(TE));
    TE_to_0_70 := exp(0.70*ln(TE));
    TE_to_0_73 := exp(0.73*ln(TE));
    TE_to_3_0  := exp(3.0*ln(TE));
    TE_to_4_5  := exp(4.5*ln(TE));

    TG_to_0_167 := exp(0.167*ln(TG));
    TG_to_0_50  := exp(0.50*ln(TG));

{Rate constants}

R1 := BQ*f1*(TE_to_0_50/sqr(ES4-ES2))
      *(1+(e*(ES4-ES2)/(k*TE)))*EXP(-(e*(ES4-ES2))/(k*TE));
```

```
R2 := BQ*f2*(TE_to_0_50/sqr(ES7-ES2))
      *(1+(e*(ES7-ES2)/(k*TE)))*EXP(-(e*(ES7-ES2))/(k*TE));
R3 := BQ*f3*(TE_to_0_50/sqr(ES4-ES6))
      *(1+(e*(ES4-ES6)/(k*TE)))*EXP(-(e*(ES4-ES6))/(k*TE));
R4 := BQ*f4*(TE_to_0_50/sqr(ES4-ES8))
      *(1+(e*(ES4-ES8)/(k*TE)))*EXP(-(e*(ES4-ES8))/(k*TE));
R5 := BQ*f5*(TE_to_0_50/sqr(ES7-ES8))
      *(1+(e*(ES7-ES8)/(k*TE)))*EXP(-(e*(ES7-ES8))/(k*TE));
R6 := BQ*f6*(TE_to_0_50/sqr(ES5-ES4))
      *(1+(e*(ES5-ES4)/(k*TE)))*EXP(-(e*(ES5-ES4))/(k*TE));
R7 := BQ*f7*(TE_to_0_50/sqr(ES5-ES7))
      *(1+(e*(ES5-ES7)/(k*TE)))*EXP(-(e*(ES5-ES7))/(k*TE));
R157:=BQ*f157*(TE_to_0_50/sqr(ES0-ES4))
      *(1+(e*(ES0-ES4)/(k*TE)))*EXP(-(e*(ES0-ES4))/(k*TE));
R8 := BQ*f8*(TE_to_0_50/sqr(ES0-ES5))
      *(1+(e*(ES0-ES5)/(k*TE)))*EXP(-(e*(ES0-ES5))/(k*TE));
R9 := BQ*f9*(TE_to_0_50/sqr(ES9-ES1))
      *(1+(e*(ES9-ES1)/(k*TE)))*EXP(-(e*(ES9-ES1))/(k*TE));

R10 := (gS2/gS4)*BQ*(f1/sqr(ES4-ES2))*TE_to_0_50*(1+(e*(ES4-ES2)/(k*TE)));
R11 := (gS2/gS7)*BQ*(f2/sqr(ES7-ES2))*TE_to_0_50*(1+(e*(ES7-ES2)/(k*TE)));
R12 := (gS6/gS4)*BQ*(f3/sqr(ES4-ES6))*TE_to_0_50*(1+(e*(ES4-ES6)/(k*TE)));
R13 := (gS8/gS4)*BQ*(f4/sqr(ES4-ES8))*TE_to_0_50*(1+(e*(ES4-ES8)/(k*TE)));
R14 := (gS8/gS7)*BQ*(f5/sqr(ES7-ES8))*TE_to_0_50*(1+(e*(ES7-ES8)/(k*TE)));
R15 := (gS4/gS5)*BQ*(f6/sqr(ES5-ES4))*TE_to_0_50*(1+(e*(ES5-ES4)/(k*TE)));
R16 := (gS7/gS5)*BQ*(f7/sqr(ES5-ES7))*TE_to_0_50*(1+(e*(ES5-ES7)/(k*TE)));
R17 := (gS5/gS0)*BQ*(f8/sqr(ES0-ES5))*TE_to_0_50*(1+(e*(ES0-ES5)/(k*TE)));
R18 := (gS1/gS9)*BQ*(f9/sqr(ES9-ES1))*TE_to_0_50*(1+(e*(ES9-ES1)/(k*TE)));

R19 := AQ*e19*TE_to_0_50*(1+(e*(ES2-ES1)/(k*TE)))
       *EXP(-(e*(ES2-ES1))/(k*TE));
R20 := AQ*e20*TE_to_0_50*(1+(e*(ES2-ES9)/(k*TE)))
       *EXP(-(e*(ES2-ES9))/(k*TE));
R151:= AQ*e151*TE_to_0_50*(1+(e*(ES3-ES1)/(k*TE)))
       *EXP(-(e*(ES3-ES1))/(k*TE));
R21 := AQ*e21*TE_to_0_50*(1+(e*(ES3-ES2)/(k*TE)))
       *EXP(-(e*(ES3-ES2))/(k*TE));
R22 := AQ*e22*TE_to_0_50*(1+(e*(ES3-ES0)/(k*TE)))
       *EXP(-(e*(ES3-ES0))/(k*TE));

R23 := 14.4E-8*CL2*(1/TE_to_4_5);
R24 := 1.8E-8*CL1*(1/TE_to_4_5);
{R25 := 0.0;
R26 := 0.0;                }

R27 := 0.0;
R28 := 0.0;

R29 := A29;
R30 := A30;
R31 := A31;
R32 := A32;
R33 := A33;

{R34 := B34;               }
```

```
R35 := 0.1*AQ*e35*TE_to_0_50*(1+(e*(EH3-EH1)/(k*TE)))
       *EXP(-(e*(EH3-EH1))/(k*TE));
R36 := 0.1*AQ*e36*TE_to_0_50*(1+(e*(EH4-EH1)/(k*TE)))
       *EXP(-(e*(EH4-EH1))/(k*TE));
R37 := AQ*e37*TE_to_0_50*(1+(e*(EH4-EH3)/(k*TE)))
       *EXP(-(e*(EH4-EH3))/(k*TE));

R38 := 0.1*(gH3/gH1)*AQ*e35*TE_to_0_50*(1+(e*(EH3-EH1)/(k*TE)));
R39 := 0.1*(gH4/gH1)*AQ*e36*TE_to_0_50*(1+(e*(EH4-EH1)/(k*TE)));
R40 := (gH4/gH3)*AQ*e37*TE_to_0_50*(1+(e*(EH4-EH3)/(k*TE)));
R41 := 3.0E-7;

R42 := 0.1*AQ*e42*TE_to_0_50*(1+(e*(EH2-EH1)/(k*TE)))
       *EXP(-(e*(EH2-EH1))/(k*TE));
R43 := AQ*e43*TE_to_0_50*(1+(e*(EH2-EH3)/(k*TE)))
       *EXP(-(e*(EH2-EH3))/(k*TE));

{R44 := 0.0;                }
R45 := AQ*e45*TE_to_0_50*(1+(e*(EH7-EH2)/(k*TE)))
       *EXP(-(e*(EH7-EH2))/(k*TE));

R46 := 0.0;
R47 := 0.0;
R48 := 9.96E-9/TE_to_4_5;
R49 := 6.0E-20;
R50 := 2.53E3/TE_to_4_5;

R51 := 0.0;
R52 := 0.0;
R53 := 2.28E-10/TE_to_0_70;
R54 := 5.14E-10/TE_to_0_50;

R55 := 0.0;
R56 := 0.0;
R57 := 3.46E-16*TG_to_0_50;
R58 := 7.1E-8*TG_to_0_167;
R59 := 0.0;
R60 := 0.0;
R61 := 0.0;
R62 := 0.0;
R63 := 0.0;
R64 := 1.0E-31;
R65 := 0.0;
R66 := 1.0E-10;
R67 := 0.0;
R68 := 2.34E-21*TE_to_0_50;

R69 := 0.0;
R70 := 0.0;
R71 := 0.0;

R72 := 0.0;
R73 := 0.0;
R74 := 0.0;
```

```
R75 := 0.0;
R76 := 0.0;

R77 := 0.1*AQ*e77*TE_to_0_50*(1+(e*(EN3-EN1)/(k*TE)))
       *EXP(-(e*(EN3-EN1))/(k*TE));
R78 := 0.1*AQ*e81*TE_to_0_50*(1+(e*(EN4-EN1)/(k*TE)))
       *EXP(-(2*(e*(EN4-EN1)))/(k*TE))*(gN4/gN1);
R79 := AQ*e82*TE_to_0_50*(1+(e*(EN4-EN3)/(k*TE)))
       *EXP(-(2*(e*(EN4-EN3)))/(k*TE))*(gN4/gN3);

R80 := 0.1*(gN3/gN1)*AQ*e77*TE_to_0_50*(1+(e*(EN3-EN1)/(k*TE)));
R81 := 0.1*sqr(gN4/gN1)*AQ*e81*TE_to_0_50*(1+(e*(EN4-EN1)/(k*TE)));
R82 := 0.1*sqr(gN4/gN3)*AQ*e82*TE_to_0_50*(1+(e*(EN4-EN3)/(k*TE)));
R83 := 3.0E-7;

R84 := 0.1*AQ*e84*TE_to_0_50*(1+(e*(EN2-EN1)/(k*TE)))
       *EXP(-(e*(EN2-EN1))/(k*TE));
R85 := AQ*e85*TE_to_0_50*(1+(e*(EN2-EN3)/(k*TE)))
       *EXP(-(e*(EN2-EN3))/(k*TE));

R86 := 0.0;
R87 := AQ*e87*TE_to_0_50*(1+(e*(EN7-EN2)/(k*TE)))
       *EXP(-(e*(EN7-EN2))/(k*TE));

R88 := 0.0;
R89 := 0.0;
R90 := 9.96E-9/TE_to_4_5;
R91 := 0.0;
R92 := 0.0;
R93 := 0.0;
R94 := 3.7E-8/TE_to_0_43;
R95 := 0.0;
R96 := 4.1E-34;
R97 := 7.0E-11;
R98 := 0.0;
R99 := 0.0;
R100 := 0.0;
R101 := 4.4E-32;
R102 := A102;
R103 := A103;
R104 := 0.0;
R105 := 0.0;
R106 := 0.0;
R107 := 0.0;
R108 := 0.0;

R109 := 1E-20;
R110 := 0.0;
R111 := 0.0;
R112 := 0.0;
R113 := 1E-20;
R114 := 0.0;

R115 := 1E-20;
R116 := 0.0;
```

```
R117 := 0.0;
R118 := 0.0;
R119 := 1E-20;
R120 := 0.0;

R121 := 6.46E-11;
R122 := 1.9E-14;
R123 := 0.0;
R124 := 0.0;
R125 := 2.53E-10;
R126 := 0.0;
R127 := 0.0;
R128 := 0.0;
R129 := 3.0E-31;
R130 := 0.0;
R131 := 2.1E-32;
R132 := 0.0;
R133 := 3.0E-11;

R134 := 0.0;
R135 := 0.0;
R136 := 0.0;

end;                    {End of code section for RATE_CONSTANTS}


PROCEDURE RATE_EQUATIONS;

{This procedure is called from SIMULATE each time-step, and sets up the system
of rate equations at time t, based on the electron/heavy particle temperature
and electric field strength
}

begin                   {Start of code section for RATE_EQUATIONS}

{First time derivatives - rate equations}

DNET := + (R19*S1*NE) + (R21*S2*NE)-(R23*S3*NE*NE)-(R24*S2*NE*NE);
  DNET := DNET + (R42*H1*NE) + (R43*H3*NE) - (R48*H2*NE*NE);
  DNET := DNET + (R20*S9*NE) + (R22*S0*NE)-(R25*S3*NE)-(R26*S2*NE);
  DNET := DNET + (R45*H2*NE) - (R50*H5*NE*NE) - (R53*H2*NE);
  DNET := DNET + (R58*H3*H3) + (R61*H1*H2) + (2*(R62*H1*H2));
  DNET := DNET - (R49*H6*NE*NE) - (R54*H5*NE) + (R56*H1*H1*H3);
  DNET := DNET + (R109*H2*S1) + (2*(R113*H3*S1)) + (2*(R151*S1*NE));
  DNET := DNET + (R84*N1*NE) + (R85*N3*NE) - (R90*N2*NE*NE);
  DNET := DNET + (R115*N2*S1) + (2*(R119*N3*S1));
  DNET := DNET - (R68*H5*H1*NE) + (R87*N2*NE) - (R94*N5*NE);
  DNET := DNET + (R98*N3*N3) + (R127*H2*N1);
DS1T := - (R19*S1*NE) - (R109*H2*S1) - (R113*H3*S1) - (R151*S1*NE);
  DS1T := DS1T - (R9*S1*NE) + (R18*S9*NE);
DS2T := - (R1*S2*NE) - (R2*S2*NE) + (R10*S4*NE) + (R11*S7*NE);
  DS2T := DS2T + (R19*S1*NE) - (R21*S2*NE) - (R24*S2*NE*NE);
  DS2T := DS2T + (R20*S9*NE) - (R26*S2*NE) + (R31*S4) + (R32*S7);
DS3T := + (R21*S2*NE) - (R23*S3*NE*NE) + (R109*H2*S1) + (R113*H3*S1);
  DS3T := DS3T + (R151*S1*NE) + (R22*S0*NE) - (R25*S3*NE);
```

```
DS4T := + (R1*S2*NE) + (R3*S6*NE) + (R4*S8*NE) - (R6*S4*NE) - (R10*S4*NE);
  DS4T := DS4T - (R12*S4*NE) - (R13*S4*NE)+(R15*S5*NE)-(R157*S4*NE);
  DS4T := DS4T + (R29*S5) - (R31*S4) - (R33*S4);
DS5T := + (R6*S4*NE) + (R7*S7*NE) - (R15*S5*NE) - (R16*S5*NE);
  DS5T := DS5T + (R17*S0*NE) - (R8*S5*NE) + (R17*S0*NE) - (R29*S5);
  DS5T := DS5T - (R30*S5);
DS6T := - (R3*S6*NE) + (R12*S4*NE) + (R33*S4);
DS7T := + (R2*S2*NE) + (R5*S8*NE) - (R7*S7*NE) - (R11*S7*NE);
  DS7T := DS7T - (R14*S7*NE) + (R16*S5*NE) + (R30*S5) - (R32*S7);
DS8T := - (R4*S8*NE) - (R5*S8*NE) + (R13*S4*NE) + (R14*S7*NE);
DS9T := + (R24*S2*NE*NE) + (R9*S1*NE) - (R18*S9*NE) - (R20*S9*NE);
  DS9T := DS9T + (R26*S2*NE);
DS0T := + (R23*S3*NE*NE) - (R17*S0*NE) + (R157*S4*NE);
  DS0T := DS0T + (R8*S5*NE) - (R17*S0*NE)-(R22*S0*NE)+(R25*S3*NE);
DH1T := - (R35*H1*NE) + (R38*H3*NE) - (R42*H1*NE) + (R109*H2*S1);
  DH1T := DH1T + (R113*H3*S1) - (R36*H1*NE) + (R39*H4*NE);
  DH1T := DH1T + (2*(R41*H6*NE)) + (2*(R54*H5*NE)) - (R56*H1*H1*H3);
  DH1T := DH1T + (R49*H6*NE*NE)
  DH1T := DH1T + (R57*H1*H3) + (R58*H3*H3) - (R61*H1*H2);
  DH1T := DH1T - (R62*H1*H2) - (R64*H1*H1*H2) + (R69*H4);
  DH1T := DH1T - (R125*H1*N4) + (R126*H2*N1) + (2*(R128*H5*N1));
  DH1T := DH1T + (R121*H3*N1) - (R129*H1*N1*N2) - (R131*H1*N1*N2);
DH2T := + (R42*H1*NE) + (R43*H3*NE) - (R48*H2*NE*NE) - (R109*H2*S1);
  DH2T := DH2T - (R45*H2*NE) - (R52*H2*NE)+(R58*H3*H3)+(R61*H1*H2);
  DH2T := DH2T - (R64*H1*H1*H2) - (R126*H2*N1);
DH3T := + (R35*H1*NE) - (R38*H3*NE) - (R43*H3*NE) - (R113*H3*S1);
  DH3T := DH3T - (R37*H3*NE) + (R40*H4*NE) - (R56*H1*H1*H3);
  DH3T := - (R57*H1*H3) - (R66*H1*H3*H3);
  DH3T := DH3T + (R70*H4) - (R121*H3*N1) + (R125*H1*N4);
DH4T := + (R48*H2*NE*NE) + (R36*H1*NE) + (R37*H3*NE) - (R39*H4*NE);
  DH4T := DH4T - (R40*H4*NE) + (R52*H2*NE) - (R69*H4) - (R70*H4);
  DH4T := DH4T + (R49*H6*NE*NE)
DH5T :=  - (R50*H5*NE*NE) - (R54*H5*NE) + (R64*H1*H1*H2);
  DH5T := DH5T + (R66*H1*H3*H3) - (R68*H5*H1*NE) - (R128*H5*N1);
DH6T :=  - (R41*H6*NE) + (R50*H5*NE*NE) + (R56*H1*H1*H3);
  DH6T := DH6T + (R68*H5*H1*NE) - (R71*H6) - (R49*H6*NE*NE);
DH7T :=  + (R45*H2*NE) + (R62*H1*H2);
DN1T := - (R77*N1*NE) + (R80*N3*NE) - (R84*N1*NE) + (R115*N2*S1);
  DN1T := DN1T + (R119*N3*S1) - (R78*N1*NE) + (R81*N4*NE);
  DN1T := DN1T + (2*(R83*N6*NE)) + (2*(R94*N5*NE)) - (R96*N1*N1*N3);
  DN1T := DN1T - (R101*N2*N1*N1) + (2*(R103*N6)) - (R121*H3*N1);
  DN1T := DN1T + (R125*H1*N4) - (R126*H2*N1) - (R127*N1*H2);
  DN1T := DN1T - (R128*H5*N1);
DN2T := + (R84*N1*NE) + (R85*N3*NE) - (R90*N2*NE*NE) - (R115*N2*S1);
  DN2T := DN2T - (R87*N2*NE) - (R101*N2*N1*N1) + (R126*H2*N1);
  DN2T := DN2T + (R127*H2*N1) + (R128*H5*N1) - (R129*H1*N1*N2);
  DN2T := DN2T - (R131*H1*H1*N2);
DN3T := + (R77*N1*NE) - (R80*N3*NE) - (R85*N3*NE) - (R119*N3*S1);
  DN3T := DN3T - (R79*N3*NE) + (R82*N4*NE) - (R96*N1*N1*N3);
  DN3T := DN3T + (R97*N4*N1) - (2*(R98*N3*N3)) ;
  DN3T := DN3T + (R102*N4) - (R122*N3*H1) + (R123*H1*N4);
DN4T := + (R90*N2*NE*NE) + (R78*N1*NE) + (R79*N3*NE) - (R81*N4*NE);
  DN4T := DN4T - (R82*N4*NE) - (R97*N4*N1);
  DN4T := DN4T - (R102*N4) + (R121*H3*N1) + (R122*H1*N3);
  DN4T := DN4T - (R125*H1*N4);
```

```
DN5T := - (R123*H1*N4) - (R94*N5*NE) + (R98*N3*N3) + (R101*N2*N1*N1);
DN6T := - (R83*N6*NE) + (R96*N1*N1*N3) - (R103*N6);
DN7T := + (R87*N2*NE);

end;                    {End of code section for RATE_EQUATIONS}

PROCEDURE TIME_DERIVS;

{
This procedure is called from SIMULATE each time-step, and sets up the second
time derivatives of all species at time t, based on the first time derivatives
(rate equations) at time t and time (t-1)
}

begin                   {Start of code section for TIME_DERIVS}

{Second time derivatives}

{       DNES   := (DNET - DNET)/(inc_time*1E-12);
        DS1S   := (DS1T - DS1T)/(inc_time*1E-12);
        DS2S   := (DS2T - DS2T)/(inc_time*1E-12);
        DS3S   := (DS3T - DS3T)/(inc_time*1E-12);
        DS4S   := (DS4T - DS4T)/(inc_time*1E-12);
        DS5S   := (DS5T - DS5T)/(inc_time*1E-12);
        DS6S   := (DS6T - DS6T)/(inc_time*1E-12);
        DS7S   := (DS7T - DS7T)/(inc_time*1E-12);
        DS8S   := (DS8T - DS8T)/(inc_time*1E-12);
        DS9S   := (DS9T - DS9T)/(inc_time*1E-12);
        DS0S   := (DS0T - DS0T)/(inc_time*1E-12);
        DH1S   := (DH1T - DH1T)/(inc_time*1E-12);
        DH2S   := (DH2T - DH2T)/(inc_time*1E-12);
        DH3S   := (DH3T - DH3T)/(inc_time*1E-12);
        DH4S   := (DH4T - DH4T)/(inc_time*1E-12);
        DH5S   := (DH5T - DH5T)/(inc_time*1E-12);
        DH6S   := (DH6T - DH6T)/(inc_time*1E-12);
        DH7S   := (DH7T - DH7T)/(inc_time*1E-12);
        DN1S   := (DN1T - DN1T)/(inc_time*1E-12);
        DN2S   := (DN2T - DN2T)/(inc_time*1E-12);
        DN3S   := (DN3T - DN3T)/(inc_time*1E-12);
        DN4S   := (DN4T - DN4T)/(inc_time*1E-12);
        DN5S   := (DN5T - DN5T)/(inc_time*1E-12);
        DN6S   := (DN6T - DN6T)/(inc_time*1E-12);
        DN7S   := (DN7T - DN7T)/(inc_time*1E-12);

end;                    {End of code section for TIME_DERIVS}


PROCEDURE SPEC_SOLN;

{
This procedure is called from SIMULATE each time-step, and solves the set of
rate equations to provide the species densities
at time t. The rate constants used are those at time (t-1). The new species
densities are calculated and used to update the rate constants at time t.
The total plasma resistance and inductance are calculated.
```

```
The time is incremented and the procedure ELEC_SOLN solves the circuit
equations. The process is repeated for the duration of the simulation time.
}

var  increase                                              :real;

begin        {Start of code section for SPEC_SOLN}
     rate_constants;     {Calculates the rate constants of each process}
     rate_equations;     {Calculates the first time derivatives of each species}
     time_derivs;        {Calculates the second time derivatives of each species}

{New species densities at time t}
S1 := S1 + (DS1T*inc_time*1E-12) + (DS1S*0.5*SQR(inc_time*1E-12));
S2 := S2 + (DS2T*inc_time*1E-12) + (DS2S*0.5*SQR(inc_time*1E-12));
S3 := S3 + (DS3T*inc_time*1E-12) + (DS3S*0.5*SQR(inc_time*1E-12));
S4 := S4 + (DS4T*inc_time*1E-12) + (DS4S*0.5*SQR(inc_time*1E-12));
S5 := S5 + (DS5T*inc_time*1E-12) + (DS5S*0.5*SQR(inc_time*1E-12));
S6 := S6 + (DS6T*inc_time*1E-12) + (DS6S*0.5*SQR(inc_time*1E-12));
S7 := S7 + (DS7T*inc_time*1E-12) + (DS7S*0.5*SQR(inc_time*1E-12));
S8 := S8 + (DS8T*inc_time*1E-12) + (DS8S*0.5*SQR(inc_time*1E-12));
S9 := S9 + (DS9T*inc_time*1E-12) + (DS9S*0.5*SQR(inc_time*1E-12));
S0 := S0 + (DS0T*inc_time*1E-12) + (DS0S*0.5*SQR(inc_time*1E-12));
H1 := H1 + (DH1T*inc_time*1E-12) + (DH1S*0.5*SQR(inc_time*1E-12));
H2 := H2 + (DH2T*inc_time*1E-12) + (DH2S*0.5*SQR(inc_time*1E-12));
H3 := H3 + (DH3T*inc_time*1E-12) + (DH3S*0.5*SQR(inc_time*1E-12));
H4 := H4 + (DH4T*inc_time*1E-12) + (DH4S*0.5*SQR(inc_time*1E-12));
H5 := H5 + (DH5T*inc_time*1E-12) + (DH5S*0.5*SQR(inc_time*1E-12));
H6 := H6 + (DH6T*inc_time*1E-12) + (DH6S*0.5*SQR(inc_time*1E-12));
H7 := H7 + (DH7T*inc_time*1E-12) + (DH7S*0.5*SQR(inc_time*1E-12));
N1 := N1 + (DN1T*inc_time*1E-12) + (DN1S*0.5*SQR(inc_time*1E-12));
N2 := N2 + (DN2T*inc_time*1E-12) + (DN2S*0.5*SQR(inc_time*1E-12));
N3 := N3 + (DN3T*inc_time*1E-12) + (DN3S*0.5*SQR(inc_time*1E-12));
N4 := N4 + (DN4T*inc_time*1E-12) + (DN4S*0.5*SQR(inc_time*1E-12));
N5 := N5 + (DN5T*inc_time*1E-12) + (DN5S*0.5*SQR(inc_time*1E-12));
N6 := N6 + (DN6T*inc_time*1E-12) + (DN6S*0.5*SQR(inc_time*1E-12));
N7 := N7 + (DN7T*inc_time*1E-12) + (DN7S*0.5*SQR(inc_time*1E-12));
NE := NE + (DNET*inc_time*1E-12) + (DNES*0.5*SQR(inc_time*1E-12));

{New total plasma conductivity at time t}

    SP := (NE*sqr(e)*1E6)/((ME*2.30E9*PH*0.76) + (ME*4.80E8*PN*0.76));

{New total plasma resistance at time t}

    R_plasma := (amp_length*100)/(SP*AR);

{New total plasma inductance at time t}

    L_plasma := (amp_length*ME*1E-4)/(NE*sqr(e)*AR);

end;        {Of code section of SPEC_SOLN}

END.
```

```
{$B+}    {Boolean complete evaluation on}
{$S+}    {Stack checking on}
{$I+}    {I/O checking on}
{$N+}    {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT SROPTO0;          {Version 1    Created on 17th October 1990}

{$R+}    {Compiler range checking for arrays, etc.}
{$F+}    {Far call compiler directive}
{$O+}    {Overlay compiler directive}

{
SROPTO0 is an overlaid unit of SRRAD0, a third-generation, zero-dimensional
strontium vapour laser simulation.  This is version 1 of the unit which
calculates the stimulated and spontaneous emission intensities emitted from
the laser medium.
}

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Printer,
  SrInit0;


procedure opto_soln;

Implementation

{-------------------------------------------------------------------------}

{Now declare the procedures and functions}


PROCEDURE OPTO_SOLN;

{
This procedure is called from SIMULATE each time-step to evaluate the
small-signal gain and loss coefficients and saturation intensity.
}

begin                        {Start of code section for OPTO_SOLN}

{       g0 := 2.64E-16*S5;
        absn := (g0/18) + ((cav_length/amp_length)*2.96E-4*PS);
}
{First time derivatives - rate equations}
```

```
      DST := (h*sqr(vcl)/430.5E-9)*(R29*S5*10E-6*1E6);
      DIT:=DST+
((amp_length/cav_length)*(sqr(430.5E-9)*vcl/1E9)*(R29/(8*pi))*(S5-(0.5*S4))*1E6*I);
      DIT := DIT - ((vcl/(2*cav_length))*ln(1/mir_ref));
{if (round(time*1E12) mod 5000 = 0) then
begin
      writeln(lst,'I = ',I:11);
      writeln(lst,'S5-(0.5*S4) = ',(S5-(0.5*S4)):11);
      writeln(lst,'DIT = ',DIT:11);
      writeln(lst,'DST = ',DST:11);
end;}

      DP430T := + (R29*S5);
      DP416T := + (R30*S5);
      DP407T := + (R31*S4);
      DP421T := + (R32*S7);
      DP1032T := + (R33*S4);

{Second time derivatives}

{     DNpS    := (DNpT - DNpT)/(inc_time*1E-12);
      DIS     := (DIT - DIT)/(inc_time*1E-12);

      DP430S := (DP430T - DP430T)/(inc_time*1E-12);
      DP416S := (DP416T - DP416T)/(inc_time*1E-12);
      DP407S := (DP407T - DP407T)/(inc_time*1E-12);
      DP421S := (DP421T - DP421T)/(inc_time*1E-12);
      DP1032S := (DP1032T - DP1032T)/(inc_time*1E-12);
}

{Solution for stimulated and spontaneous emission intensities
}

Np   := Np + (DNpT * inc_time*1E-12) + (DNpS*0.5*SQR(inc_time*1E-12));
I    := 0.0{I + (DIT * inc_time*1E-12) + (DIS*0.5*SQR(inc_time*1E-12))};
P430:= P430 + (DP430T*inc_time*1E-12) + (DP430S*0.5*SQR(inc_time*1E-12));
P416:= P416 + (DP416T*inc_time*1E-12) + (DP416S*0.5*SQR(inc_time*1E-12));
P407:= P407 + (DP407T*inc_time*1E-12) + (DP407S*0.5*SQR(inc_time*1E-12));
P421:= P421 + (DP421T*inc_time*1E-12) + (DP421S*0.5*SQR(inc_time*1E-12));
P1032:=P1032+(DP1032T*inc_time*1E-12)+(DP1032S*0.5*SQR(inc_time*1E-12));

end;      {Of code section of OPTO_SOLN}

END.
```

```
{$B+}    {Boolean complete evaluation on}
{$S+}    {Stack checking on}
{$I+}    {I/O checking on}
{$N+}    {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT SREQNS0;          {Version 1    Created on 17th October 1990}

{$R+}    {Compiler range checking for arrays, etc.}
{$F+}    {Far call compiler directive}
{$O+}    {Overlay compiler directive}

{
SREQNS0 is an overlaid unit of SRRAD0, a third-generation, zero-dimensional
strontium vapour laser simulation.  This is version 1 of the unit which is
lists the chemical processes considered in the model.
}

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Printer,
  SrInit0;

VAR  RX                                    :integer;

{Above are the global arrays to hold amplifier, pulse train & ASE data}

procedure processes;
procedure species;

Implementation

{------------------------------------------------------------------------}

{Now declare the procedures and functions}

PROCEDURE SPECIES;


var    result                              :real;

begin              {Start of code section for SPECIES procedure}

       writeln(lst, title, cr);

{Chemical species considered in the model}

{
```

1. Strontium species

Sr	Ground state strontium I	S1
Sr+	5S1/2 Ground state strontium II	S2
Sr++	Ground state strontium III	S3
Sr1+	5P3/2 Sr (lower laser level: l= 430.5nm)	S4
Sr2+	6S1/2 Sr (upper laser level: l= 430.5nm)	S5
Srm+	4D5/2 Sr II metastable state	S6
Sr1'+	5P1/2 Sr (lower laser level: l= 416.2nm)	S7
Srm'+	4D3/2 Sr II metastable state	S8
Sr*	Strontium I pseudo-state	S9
Sr*+	Strontium II pseudo-state	S0

2. Helium species

He	Ground state atomic helium	H1
He+	Singly-ionized atomic helium	H2
Hem	Helium metastable	H3
He*	Atomic helium pseudo-state	H4
He2+	Singly-ionized molecular helium	H5
He2*	Molecular helium pseudo-state	H6
He++	Doubly-ionized atomic helium	H7

3. Neon species

Ne	Ground state atomic neon	N1
Ne+	Singly-ionized atomic neon	N2
Nem	Neon metastable	N3
Ne*	Atomic neon pseudo-state	N4
Ne2+	Singly-ionized molecular neon	N5
Ne2*	Molecular neon pseudo-state	N6
Ne++	Doubly-ionized atomic neon	N7

4. Electrons

e	Electrons	NE

5. Photons

p	Photons	I
p430	Spontaneously emitted photons @ 430nm	P430
p416	Spontaneously emitted photons @ 416nm	P416
p407	Spontaneously emitted photons @ 407nm	P407
p421	Spontaneously emitted photons @ 421nm	P421
p1032	Spontaneously emitted photons @ 1032nm	P1032

}

end; {Of code section of SPECIES}

PROCEDURE PROCESSES;

var result :real;

begin {Start of code section for PROCESSES procedure}

writeln(lst, title, cr);

{A. STRONTIUM

ELECTRON-IMPACT EXCITATION REACTIONS

Process No. 1 is Sr+ + e -> Sr1+ + e
Process No. 2 is Sr+ + e -> Sr1'+ + e
Process No. 3 is Srm+ + e -> Sr1+ + e
Process No. 4 is Srm'+ + e -> Sr1+ + e
Process No. 5 is Srm'+ + e -> Sr1'+ + e
Process No. 6 is Sr1+ + e -> Sr2+ + e
Process No. 7 is Sr1'+ + e -> Sr2+ + e
Process No.157 is Sr1+ + e -> Sr+* + e
Process No. 8 is Sr2+ + e -> Sr+* + e
Process No. 9 is Sr + e -> Sr* + e

ELECTRON-IMPACT IONIZATION REACTIONS

Process No. 19 is Sr + e -> Sr+ + e + e
Process No. 20 is Sr* + e -> Sr+ + e + e
Process No.151 is Sr + e -> Sr++ + e + e + e
Process No. 21 is Sr+ + e -> Sr++ + e + e
Process No. 22 is Sr+* + e -> Sr++ + e + e

ELECTRON-IMPACT DE-EXCITATION REACTIONS

Process No. 10 is Sr1+ + e -> Sr+ + e
Process No. 11 is Sr1'+ + e -> Sr+ + e
Process No. 12 is Sr1+ + e -> Srm+ + e
Process No. 13 is Sr1+ + e -> Srm'+ + e
Process No. 14 is Sr1'+ + e -> Srm'+ + e
Process No. 15 is Sr2+ + e -> Sr1+ + e
Process No. 16 is Sr2+ + e -> Sr1'+ + e
Process No. 17 is Sr+* + e -> Sr2+ + e
Process No. 18 is Sr* + e -> Sr + e

COLLISIONAL-RADIATIVE RECOMBINATION

Process No. 23 is Sr++ + e +e-> Sr+* + e
Process No. 24 is Sr+ + e + e-> Sr* + e
Process No. 25 is Sr++ + e -> Sr+* + hv3+*
Process No. 26 is Sr+ + e -> Sr* + hv1*

STIMULATED AND SPONTANEOUS EMISSION

Process No. 27 is Sr2+ + hv430-> Sr1+ + hv430 + hv430
(Stim. emission @ 430.5nm))

Process No. 28 is Sr2+ + hv416 -> Sr1'+ + hv416 + hv416
(Stim. emission @ 416.2nm))

Process No. 29 is Sr2+ -> Sr1+ + hv21 (Spontaneous emission @ 430.5nm)
Process No. 30 is Sr2+ -> Sr1'+ + hv21'(Emission @ 416.2nm)
Process No. 31 is Sr1+ -> Sr+ + hv10 (Emission)
Process No. 32 is Sr1'+ -> Sr+ + hv1'0 (Emission)
Process No. 33 is Sr1+ -> Srm+ + hv1m (Emission)

ABSORPTION

Process No. 34 is Reabsorption of hv10 (Reabsorption @ 430.5nm)

B. HELIUM

ELECTRON-IMPACT EXCITATION REACTIONS

Process No. 35 is He + e -> Hem + e
Process No. 36 is He + e -> He* + e
Process No. 37 is Hem + e -> He* + e

ELECTRON-IMPACT IONIZATION REACTIONS

Process No. 42 is He + e -> He+ + e + e
Process No. 43 is Hem + e -> He+ + e + e
Process No. 44 is He* + e -> He+ + e + e
Process No. 45 is He+ + e -> He++ + e + e

ELECTRON-IMPACT DE-EXCITATION REACTIONS

Process No. 38 is Hem + e -> He + e
Process No. 39 is He* + e -> He + e
Process No. 40 is He* + e -> Hem + e
Process No. 41 is He2* + e -> He + He + e

COLLISIONAL-RADIATIVE RECOMBINATION

Process No. 46 is He+ + e + e-> He + e
Process No. 47 is He+ + e + e-> Hem + e
Process No. 48 is He+ + e + e-> He* + e
Process No. 49 is He2+ + e +e-> He* + He + e
Process No. 50 is He2+ + e +e-> He2* + e

RADIATIVE RECOMBINATION

Process No. 51 is He+ + e -> He + hv
Process No. 52 is He+ + e -> Hem + hv
Process No. 53 is He+ + e -> He* + hv

DISSOCIATIVE RECOMBINATION

Process No. 54 is He2+ + e -> (He2*) ->
He* + He-> He + He + hv

ATOM-ATOM COLLISIONAL PROCESSES

Process No. 55 is He + Hem -> Hem + He
Process No. 56 is Hem +He +He-> He2* + He
Process No. 57 is He + Hem -> He + He
Process No. 58 is Hem + Hem -> He + He+ + e
Process No. 59 is He+ + He -> He + He+
Process No. 60 is He+ + Hem -> Hem + He+
Process No. 61 is He+ + He -> He+ + He+ + e
Process No. 62 is He+ + He -> He++ + He+ + e + e
Process No. 63 is He+ + He +e-> He2+ + e

Process No. 64 is He+ +He +He-> He2+ + He
Process No. 65 is Hem + Hem -> He2+ + e
Process No. 66 is Hem +Hem +H-> He2+ + He + e
Process No. 67 is He+ + He +e-> He* + He
Process No. 68 is He2+ +He +e-> He2* + He

STIMULATED AND SPONTANEOUS EMISSION

Process No. 69 is He* -> He + hv (Spontaneous emission)
Process No. 70 is He* -> Hem + hv (Spontaneous emission)
Process No. 71 is He2* -> He2*'-> He2** + hv (Spontaneous emission)

TRANSPORT PROCESSES

Process No. 72 is He -> walls
Process No. 73 is Hem -> walls
Process No. 74 is He* -> walls
Process No. 75 is He+ -> walls
Process No. 76 is He2+ -> walls

C. NEON

ELECTRON-IMPACT EXCITATION REACTIONS

Process No. 77 is Ne + e -> Nem + e
Process No. 78 is Ne + e -> Ne* + e
Process No. 79 is Nem + e -> Ne* + e

ELECTRON-IMPACT IONIZATION REACTIONS

Process No. 84 is Ne + e -> Ne+ + e + e
Process No. 85 is Nem + e -> Ne+ + e + e
Process No. 86 is Ne* + e -> Ne+ + e + e
Process No. 87 is Ne+ + e -> Ne++ + e + e

ELECTRON-IMPACT DE-EXCITATION REACTIONS

Process No. 80 is Nem + e -> Ne + e
Process No. 81 is Ne* + e -> Ne + e
Process No. 82 is Ne* + e -> Nem + e
Process No. 83 is Ne2* + e -> Ne + Ne + e

COLLISIONAL-RADIATIVE RECOMBINATION

Process No. 88 is Ne+ + e + e-> Ne + e
Process No. 89 is Ne+ + e + e-> Nem + e
Process No. 90 is Ne+ + e + e-> Ne* + e

RADIATIVE RECOMBINATION

Process No. 91 is Ne+ + e -> Ne* + hv
Process No. 92 is Ne+ + e -> Nem + hv
Process No. 93 is Ne+ + e -> Ne + hv

DISSOCIATIVE RECOMBINATION

Process No. 94 is Ne2+ + e -> (Ne2*) + hv ->
Ne* + Ne-> Ne + Ne + hv

ATOM-ATOM COLLISIONAL PROCESSES

Process No. 95 is Ne + Nem -> Nem + Ne
Process No. 96 is Nem +Ne +Ne-> Ne2* + Ne
Process No. 97 is Ne* + Ne -> Nem + Ne
Process No. 98 is Nem + Nem -> Ne + Ne+ + e
-> Ne2+ + e
Process No. 99 is Ne+ + Ne -> Ne + Ne+
Process No.100 is Ne+ + Nem -> Nem + Ne+
Process No.101 is Ne+ +Ne +Ne-> Ne2+ + Ne

STIMULATED AND SPONTANEOUS EMISSION

Process No.102 is Ne* -> Nem + hv (Spontaneous emission)
Process No.103 is Ne2* -> Ne + Ne + hv (Spontaneous emission)

ABSORPTION

Process No.104 is Ne* + hv -> Ne+ + e
Process No.105 is Ne2+ + hv -> Ne+ + Ne

TRANSPORT PROCESSES

Process No.106 is Nem -> walls
Process No.107 is Ne+ -> walls
Process No.108 is Ne2+-> walls

D. HELIUM-STRONTIUM

ATOM-ATOM COLLISIONAL PROCESSES

Process No.109 is He+ + Sr -> He + Sr++ + e
Process No.110 is He+ + Sr -> He + Sr+
Process No.111 is He+ + Sr+ -> He + Sr++
Process No.112 is He2+ + Sr -> He + He + Sr++ + e
Process No.113 is Hem + Sr -> He + Sr++ + e + e
Process No.114 is Hem + Sr+ -> He + Sr++ + e

E. NEON-STRONTIUM

ATOM-ATOM COLLISIONAL PROCESSES

Process No.115 is Ne+ + Sr -> Ne + Sr++ + e
Process No.116 is Ne+ + Sr -> Ne + Sr+
Process No.117 is Nem + Sr -> Ne + Sr++ + e + e
Process No.118 is Ne2+ + Sr -> Ne + Ne + Sr++ + e
Process No.119 is Nem + Sr -> Ne + Sr++ + e + e
Process No.120 is Nem + Sr+ -> Ne + Sr++ + e

F. HELIUM-NEON

ATOM-ATOM COLLISIONAL PROCESSES

```
      Process No.121 is   Hem + Ne   -> He + Ne*
      Process No.122 is   He + Nem   -> He + Ne*
      Process No.123 is   He + Ne*   -> He + Nem
      Process No.124 is   Hem + Ne   -> He + Ne
      Process No.125 is   He + Ne*   -> Hem + Ne
      Process No.126 is   He+ + Ne   -> He + Ne+
      Process No.127 is   He+ + Ne   -> He+ + Ne+ + e
      Process No.128 is   He2+ + Ne  -> He + He + Ne+
      Process No.129 is   He +Ne+ +Ne-> He + Ne2+
                                     -> (HeNe)+ + Ne
      Process No.130 is   Hem + Ne   -> (HeNe)+ + e
      Process No.131 is   He +He+ +Ne+->(HeNe)+ + He
      Process No.132 is   (HeNe)+ + e-> (HeNe)*
                                     -> He* + Ne*
      Process No.133 is   (HeNe)+ +Ne-> He + Ne2+
```

TRANSPORT PROCESSES

```
      Process No.134 is   Nem     -> walls (in He)
      Process No.135 is   Ne+     -> walls (in He)
      Process No.136 is   (HeNe)+-> walls (in He)
}
end;  {Of code section of PROCESSES}

END.
```

Appendix E

G. C. A. P. (Generalised Circuit Analysis Program) Program Listing

1. INTRODUCTION

GCAP is a general-purpose circuit simulation program for pulsed power. Any system that can be modelled as a collection of linear or non-linear, time-invariant or time-varying circuit elements (resistors, capacitors, inductors, independent or dependent (voltage- or current-controlled) voltage and current sources, etc.) can be analysed.

2. GCAP PROGRAM LISTING

A complete program listing of GCAP is included in this appendix. The program structure is illustrated in the flow diagram of Figures 2.2 and 2.3. The main program is known as GCAP03. The functions of the subprograms are as follows:

GCTITL:	Introduction to program
GCINFO:	On-screen information on GCAP
GCLIBR:	Library of complex circuit element models
GCUSER:	Interface to a user-defined model
GCINPT:	Input data to the program
GCINIT:	Initialisation of input parameters of current simulation
GCFILE:	Stores circuit element, analysis and format descriptor information in files
GCSORT:	Sorts the network branches into a proper tree

GCGRAP:	Graphics display of the circuit under simulation
GCMAT:	Manipulates matrix operation
GCMATQ:	Formulation of the fundamental cutset matrix Q and fundamental submatrix F
GCSTAT:	Formulation of the state matrices A, B, C, D and E
GCOPPT:	Calculation of the operating point of a circuit containing non-linear elements
GCDC:	Non-linear DC analysis
GCAC:	Linear AC analysis
GCTEMP:	Transient analysis
GCFREQ:	Frequency analysis
GCHEAD:	On-screen display of parameters in use
GCOUT1:	Tabular display of the results of a simulation
GCOUT2:	Two-dimensional graphics display of the results of a simulation
GCOUT3:	Three-dimensional graphics display of the results of a simulation

```
{$B+}   {Boolean complete evaluation on}
{$S+}   {Stack checking on}
{$I+}   {I/O checking on}
{$N+}   {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}

PROGRAM GCAP03;          {Version 3          Created on 7th May 1991}
                         {Last modified on 14th June 1991}

{$R+}   {Compiler range checking for arrays, etc.}
{$F+}   {Far call compiler directive}

{
GCAP is a Generalised Circuit Analysis Program for pulsed power, devised
and written within the Department of Physics and Astronomy at the
University of St. Andrews, St. Andrews, Fife, Scotland.
GCAP will simulate and analyse any electrical circuit.

This is version 3 of the program, which is written in Turbo Pascal Version 5.5.

The input routine has been devised to act like a screen editor with preset
default values for a demonstration program.  The format
of the output data has been written to enable the display of discharge
voltage and current waveforms as
requested.

Also included is the option of iterating various input parameters over a
user-specified range of values.

Variables used are:-

STRUCTURE
---------

        TITLE   :   Title of present GCAP simulation
        INFO    :   Information on GCAP input format/capabilities
        INFONO  :   Number of sub-topic of GCAP of interest
        MORE    :   Further information on GCAP input format/capabilities

ITERATION
---------

        IA-IP   :   Column parameter for formulating F-submatrices
        JA-JP   :   Row parameter for formulating F-submatrices
        I1      :   Parameter for formulating F-matrix
        J       :   Parameter for formulating F-matrix
        ROW     :
        COL     :
```

COMPONENT NUMBERS

W	:	Number of tree branch independent voltage sources
DD	:	Number of tree branch capacitances
BB	:	Number of tree branch conductances
EE	:	Number of tree branch reciprocal inductances
U	:	Number of link reciprocal capacitances
X	:	Number of link resistances
V	:	Number of link inductances
AA	:	Number of link independent current sources
WX	:	Number of independent voltage sources for consistency check
DDX	:	Number of capacitances for consistency check
XX	:	Number of resistances for consistency check
VX	:	Number of inductances for consistency check
AAX	:	Number of independent current sources for consistency check
N	:	Number of nodes
B1	:	Number of branches
T1	:	N-1
P	:	B1-T1
H	:	J-T1
AX	:	DD+V
BX	:	W+AA
XB	:	U+1
XC	:	U+X+1
XD	:	U+X+V+1
WB	:	U+X
WC	:	U+X+V
WD	:	U+X+V+AA
MB	:	JB-U
MC	:	JC-XC+1
MD	:	JD-XD+1
YE	:	W+1
YI	:	W+DD+1
YM	:	W+DD+BB+1
Y1	:	W+DD
Y2	:	W+DD+BB
Y3	:	W+DD+BB+EE
NE	:	IE-W
NF	:	I6-W
NG	:	IG-YE+1
NH	:	IH-W
NI	:	II-Y1

NJ	:	IJ-Y1
NK	:	IK-YI+1
NL	:	IL-Y1
NM	:	IM-Y2
NN	:	IN-Y2
NO	:	IO-Y2
NP	:	IP-Y2
MF	:	JF-U
MG	:	JG-XC+1
MH	:	JH-XD+1
MJ	:	JJ-U
MK	:	JK-XC+1
ML	:	JL-XD+1
MN	:	JN-U
MO	:	JO-XC+1
MP	:	JP-XD+1
Z1	:	W+DD+BB+EE
Z2	:	U+X+V+AA
NX	:	Parameter for compiling COMPON array

MATRICES

Q	:	Fundamental cutset matrix
F	:	Fundamental F matrix
FA	:	F-submatrix Fvs
FB	:	F-submatrix Fvr
FC	:	F-submatrix Fvl
FD	:	F-submatrix Fvi
FE	:	F-submatrix Fcs
F6	:	F-submatrix Fcr
FG	:	F-submatrix Fcl
FH	:	F-submatrix Fci
FI	:	F-submatrix Fgs
FJ	:	F-submatrix Fgr
FK	:	F-submatrix Fgl
FL	:	F-submatrix Fgi
FM	:	F-submatrix Fts
FQ	:	F-submatrix Ftr
FO	:	F-submatrix Ftl
FP	:	F-submatrix Fti
HA	:	F-submatrix F'vs
HB	:	F-submatrix F'vr
HC	:	F-submatrix F'vl
HD	:	F-submatrix F'vi
HE	:	F-submatrix F'cs
HF	:	F-submatrix F'cr
HG	:	F-submatrix F'cl
HH	:	F-submatrix F'ci
HJ	:	F-submatrix F'gr
HK	:	F-submatrix F'gl

HL	:	F-submatrix F'gi
HO	:	F-submatrix F'tl
HP	:	F-submatrix F'ti
CC	:	Capacitance matrix Cc
CS	:	Elastance matrix Cs
GG	:	Conductance matrix Gg
RR	:	Resistance matrix Rr
LL	:	Inductance matrix Ll
LT	:	Reciprocal inductance matrix Lt
RI	:	Inverse of matrix Rr
GI	:	Inverse of matrix Gg
UA	:	Intermediate matrix
UB	:	Intermediate matrix
UC	:	Intermediate matrix
UD	:	Intermediate matrix
UE	:	Intermediate matrix
UF	:	Intermediate matrix
UG	:	Intermediate matrix
UH	:	Intermediate matrix
CA	:	Capacitance matrix C
L	:	Inductance matrix L
R	:	Resistance matrix R
G	:	Conductance matrix G
RJ	:	Inverse of matrix R
GJ	:	Inverse of matrix G
CJ	:	Inverse of matrix C
LJ	:	Inverse of matrix L
H1	:	Intermediate matrix
H2	:	Intermediate matrix
H3	:	Intermediate matrix
H4	:	Intermediate matrix
H5	:	Intermediate matrix
H6	:	Intermediate matrix
H7	:	Intermediate matrix
H8	:	Intermediate matrix
H0	:	Intermediate matrix
W2	:	Intermediate matrix
W3	:	Intermediate matrix
W5	:	Intermediate matrix
W7	:	Intermediate matrix
W8	:	Intermediate matrix
W9	:	Intermediate matrix
W0	:	Intermediate matrix
YU	:	Intermediate matrix
YV	:	Intermediate matrix
YW	:	Intermediate matrix
Y1	:	Intermediate matrix

Y2	:	Intermediate matrix
Y3	:	Intermediate matrix
Y4	:	Intermediate matrix
H9	:	Hybrid matrix Hcc
Y5	:	Hybrid matrix Hcl
W1	:	Hybrid matrix Hcv
Y6	:	Hybrid matrix Hci
Y7	:	Hybrid matrix Hlc
W4	:	Hybrid matrix Hll
Y8	:	Hybrid matrix Hlv
W6	:	Hybrid matrix Hli

General subprograms used are:-

(c) = Automatically performed by GCAP
(o) = Performed by GCAP only if specified by user
(a) = Performed by GCAP if referenced in program execution

GENERAL EXECUTION

START1	:	
INPUT1	:	
FILES1	:	

GCAP INFORMATION

INFY	:	Print information on GCAP	(o)
GENI	:	GCAP General information	(o)
CAPB	:	GCAP Capabilities	(o)
STRU	:	GCAP Structure	(o)
FORM	:	State equation Formulation	(o)
SOLU	:	State equation Solution	(o)
ACCE	:	Acceptable circuit components	(o)
NONL	:	Nonlinear modelling	(o)
TIMV	:	Time-varying modelling	(o)
USER	:	User-defined models	(o)
LIBR	:	Library of GCAP-defined models	(o)
INP1	:	Input file COMPON	(o)
INP2	:	Input file ANALYS	(o)
INP3	:	Input file STRUCT	(o)
MORY	:	More information on GCAP	(o)
ENTER	:	Prints instructions to enter data for all three input files	(c)

INPUT FILE "COMPON" MANIPULATION

COMP	:	Stores circuit topology data in COMPON (COMP1 and COMP2)	(c)

COUNT	:	Counts and displays the number and type of each element in the circuit (o)
CONI	:	Performs a consistency check on total numbers of each type of element entered (o)
FILES	:	Prints out each input file (o)
SORT	:	Sorts circuit branches into a proper tree (c)
SORT2	:	Calculates number of nodes aswell as maximum and minimum node numbers (c)
FIND	:	Identifies in turn independent voltage sources, capacitances, resistances, inductances and independent current sources (c)
NODDUP	:	Ensures that no two nodes are surplus to requirements via a particular element type (c)
ORDER	:	Places nodes in ascending order (c)
ELIM	:	Eliminates surplus nodes (c)
NODCOV	:	Ensures that all nodes are covered (c)
NODCON	:	Ensures that all nodes are connected (c)
DIMENS	:	Takes SORT and calculates the dimensions of the F submatrices (c)
CONNEC	:	Accepts the data from SORT to connect the branches for use by MATQ (c)
GRAP	:	Creates a connected graph of the circuit for display (c)
MATQ	:	Calculates the fundamental cutset matrix Q from the array COMPON and the proper tree (c)
PRMATQ	:	Prints out the fundamental cutset matrix Q (o)
MATF	:	Calculates the fundamental F submatrix from the fundamental cutset matrix Q (c)
PRMATF	:	Prints out the fundamental F submatrix (o)
FSUB	:	Formulates the F submatrices by partitioning the fundamental F submatrix (c)
PRFSUB	:	Prints out the F submatrices (o)
ELEM	:	Calculates the element matrices Cc, Cs, Gg, Rr, Lt and Ll (c)
INV1	:	Calculates the inverse matrices Rr-1 and Gg-1 (c)
INT1	:	Calculates the intermediate matrices UA -> UH and matrices C, L, R and G (c)
INV2	:	Calculates the inverse matrices R-1, G-1, C-1 and L-1 (c)
HYBR	:	Calculates the hybrid matrices Hcc, Hcl, etc. (c)
STAT	:	Calculates the state matrices A, B, C, D and E (c)

INPUT FILE "ANALYS" MANIPULATION

ANLY	:	Stores analysis specification data in ANALYS

(ANLY1 and ANLY2) (c)

TIME1 : Sets time increment and interval of calculation (c)

INPUT FILE "STRUCT" MANIPULATION

STRC : Stores GCAP output formatting data in STRUCT (STRC1 and STRC2) (c)

TITL : Assigns a title to each section of output by scanning input file STRC1 (c)

PRTITL : Prints out default title (o)

PTITL2 : Prints out assigned title (o)

INFO : Calls subprogram INFY to display information options available by scanning input file STRC1 (o)

PRIMAT : Prints out all matrices (o)

PRINT : Prints out in tabular form the specified circuit parameters (o)

PLOT2 : Plots a two-dimensional output of the specified circuit parameters (o)

PLOT3 : Plots a three-dimensional output of the specified circuit parameters (o)

MATRIX MANIPULATION

MATMUL : General matrix multiplication subprogram to calculate the product of two matrices which can be used with matrices of any dimensions (a)

MATPRI : General matrix printout subprogram which can be used with matrices of any dimensions. MATPRI is called by PRIMAT (a)

MATPLU : General matrix addition subprogram for two matrices which can be used with matrices of any dimensions (a)

MATMIN : General matrix subtraction subprogram for two matrices which can be used with matrices of any dimensions (a)

MATTRN : General subprogram to determine the transpose of a matrix which can be used with matrices of any dimensions (a)

MATINV : General subprogram to determine the inverse of a matrix which can be used with matrices of any dimensions (a)

MATIDN : General subprogram to equate a matrix to the identity matrix which can be used with matrices of any dimensions (a)

MATZER : General subprogram to equate a matrix to the zero matrix which can be used with matrices of any dimensions (a)

MATPRD : General subprogram to calculate the product of

a constant with a matrix which can be used with matrices of any dimensions (a)

MATMCH : General subprogram to equate one matrix to a second matrix which can be used with matrices of any dimensions (a)

STATE EQUATION SOLUTION

INITIL : Initialises all state matrices for Runge-Kutta solution (c)

RUNGEK : Solution of the state equations by means of a Runge-Kutta algorithm (c)

}

{Start of global type, constant and variable declarations}

```
Uses
  Overlay,
  Crt,
  Printer,
  GcTitl,
  GcInfo,
  GcLibr,
  GcUser,
  GcInpt,
  GcInit,
  GcFile,
  GcSort,
  GcGrap,
  GcMat,
  GcMatq,
  GcStat,
  GcOppt,
  GcDC,
  GcAC,
  GcTemp,
  GcFreq,
  GcHead,
  GcOut1,
  GcOut2,
  GcOut3{,
  GcEqns};

{$O GcTitl}
{$O GcInfo}
{$O GcLibr}
{$O GcUser}
{$O GcInpt}
{$O GcInit}
{$O GcFile}
{$O GcSort}
{$O GcGrap}
```

```
{$O GcMat}
{$O GcMatq}
{$O GcStat}
{$O GcOppt}
{$O GcDC}
{$O GcAC}
{$O GcTemp}
{$O GcFreq}
{$O GcHead}
{$O GcOut1}
{$O GcOut2}
{$O GcOut3}

var     start                           :integer;

{-------------------------------------------------------------------------}

{Now declare the procedures and functions}

PROCEDURE INSTALL;

{Initialisation of Turbo Pascal's Overlay Manager}

const       OvrMaxSize = 8;

var         OvrName:            string[79];
            Size:               LongInt;

begin  {Start of initialisation of Turbo Pascal's Overlay Manager}

   OvrName:='GCAP03.OVR';
   repeat
      OvrInit (OvrName);
      if OvrResult=ovrNotFound then
      begin
         writeln('Overlay file not found: ',OvrName,'.');
         write('Enter correct overlay file name: ');
         readln(OvrName);
      end;
   until OvrResult<>ovrNotFound;

   if OvrResult<>ovrOK then
   begin
      writeln('Overlay manager error.');
      Halt(1);
   end;
   OvrInitEMS;
   if OvrResult<>OvrOK then
   begin
      case OvrResult of
         ovrIOError:          write('Overlay file I/O error');
         ovrNoEMSDriver:      write('EMS driver not installed');
         ovrNoEMSMemory: write('Not enough EMS memory');
   end;
   write('.Press Enter...');
```

```
   readln;
   end;
   OvrSetBuf(OvrMaxSize);
end;        {End of initialisation of Turbo Pascal's Overlay Manager}

PROCEDURE CONSTANT;

{
 Procedure to calculate the specific constants of the system
}

var   const1, const2                                    :real;

begin

      const1  :=  4*pi;
      const2  :=  h*vcl;

end;

PROCEDURE DECIDA;

{
 Decision on analysis type to be performed by GCAP
}
var   option1          :string[6];
      num              :integer;

begin

{             clrscr;
              gotoxy(1,10);
              writeln(sp10,'Enter an analysis type:');
              gotoxy(1,12);
              writeln(sp18,'DC. DC analysis');
              gotoxy(1,14);
              writeln(sp18,'AC. AC analysis');
              gotoxy(1,16);
              writeln(sp18,'TRAN. Transient analysis');
              gotoxy(1,18);
              writeln(sp18,'FREQ. Frequency analysis');

              readln(option1);                          }

{             if option1 = 'DC' then
                  nonlindc    }     {Main nonlinear DC simulation procedure}
{             else if option1 = 'AC' then
                  linearac    }     {Main linear AC simulation procedure}
{             else if option1 = 'TRAN' then
                  nonlintra   }     {Main nonlinear transient simulation procedure}
{             else if option1 = 'FREQ' then
                  frequency;  }     {Main frequency simulation procedure}

              for num := 1 to num2 do
              begin
```

```
                if ANLY1[num,1] = 'DC' then
                   nonlindc       {Main nonlinear DC simulation procedure}
                else if ANLY1[num,1] = 'AC' then
                   linearac       {Main linear AC simulation procedure}
                else if ANLY1[num,1] = 'TRAN' then
                   nonlintra      {Main nonlinear transient simulation procedure}
                else if ANLY1[num,1] = 'FREQ' then
                   frequency ;    {Main frequency simulation procedure}
             end;

end;

PROCEDURE DECIDS;

{
 Decision on format of output from GCAP
}
var   option2        :string[6];
      num            :integer;

begin

{              clrscr;
             gotoxy(1,10);
             writeln(sp10,'Enter form of output required:');
             gotoxy(1,12);
             writeln(sp18,'Table. Tabular output');
             gotoxy(1,14);
             writeln(sp18,'2dgraph. Two-dimensional graphics');
             gotoxy(1,16);
             writeln(sp18,'3dgraph. Three-dimensional graphics');

             readln(option2);          }

{              if option2 = 'table' then
                 table    }          {Output of tabular data to printer}
{              else if option2 = '2dgraph' then
                 twodgraph}          {Output of 2-dimensional plotted data to printer}
{              else if option2 = '3dgraph' then
                 threedgraph; }      {Output of 3-dimensional plotted data to printer}

             for num := 1 to num3 do
             begin
                if STRC1[num,1] = 'table' then
                table    }          {Output of tabular data to printer}
                else if STRC1[num,1] = '2dgraph' then
                twodgraph}          {Output of 2-dimensional plotted data to printer}
                else if STRC1[num,1] = '3dgraph' then
                threedgraph; }      {Output of 3-dimensional plotted data to printer}
             end;

end;


{-------------------------------------------------------------------------}
```

```
{This is the main program, Version 1, created on May 7th, 1991}

BEGIN
     install;
     start1;                    {Produces a title screen}
{     readln (start);
     if start = 1 then
        info
     else if start = 3 then
        initialise
     else if start = 4 then halt;       }
     enter;
     while true do              {Endless loop: terminates by EXIT, 4 lines down}
     begin
{          initialise;       }     {Read in starting values}
{          run := ask_input (one_way);            }
{          if not run then exit;       }  {This is the escape from the WHILE loop}
             files;
             constant;                  {Calculates constants used in the code}
             sort;
             graphics;                  {Provides an on-screen graphic display of
                                          circuit}
             matrixq;
             inittv;
             statem;
             dcoppt;

             decida;                    {Decision on analysis type}

             header;                    {Provides an on-screen progress of
                                         simulation}

             decids;                    {Decision on format of output}

             {wrapup;}                  {Saves files to disk if required}
     end;                               {Of WHILE loop}
END.
```

```
{$B+}    {Boolean complete evaluation on}
{$S+}    {Stack checking on}
{$I+}    {I/O checking on}
{$N+}    {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT GCTITL;          {Version 1      Created on 12th June 1991}

{$R+}    {Compiler range checking for arrays, etc.}
{$F+}    {Far call compiler directive}
{$O+}    {Overlay compiler directive}


{
 GCTITL is an overlaid unit of GCAP03, a Generalised Circuit Analysis Program
 for pulsed power.  This is version 1 of the unit which is used to introduce
 the user to the program
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Printer,
  GcInit,
  GcInfo;

procedure start1;

Implementation

{-------------------------------------------------------------------------}

{Now declare the procedures and functions}

PROCEDURE START1;

var   inp                  :string[1];

begin

   clrscr;
   gotoxy(1,2);
   writeln(sp10,'G.C.A.P. : GENERALISED CIRCUIT ANALYSIS PROGRAM');
   gotoxy(1,4);
      writeln(sp14,'Computer Modelling of Electrical Circuits');
   gotoxy(1,5);
      writeln(sp21,'by State-Variable Analysis');
   gotoxy(1,7);
      writeln(sp10,'Compiled by:');
```

```
    gotoxy(1,8);
       writeln(sp25,'Andrew K. Kidd');
    gotoxy(1,9);
       writeln(sp25,'University of St. Andrews');
    gotoxy(1,10);
       writeln(sp25,'Fife, Scotland');
    gotoxy(1,11);
       writeln(sp25,'KY16 9SS');
    gotoxy(1,12);
       writeln(sp25,'Tel: (0334) 76161 ext. 8307/8368/8338');
    gotoxy(1,14);
       writeln(sp10,'Date: September 1987 - October 1988');
    gotoxy(1,16);
       writeln(sp18,'C. Continue simulation');
    gotoxy(1,18);
       writeln(sp18,'I. Information');
    gotoxy(1,20);
       writeln(sp18,'D. Demonstration program');
    gotoxy(1,22);
       writeln(sp18,'Q. Quit GCAP');
    gotoxy(1,24);
       writeln(sp10,'Enter a parameter:  Press any key to continue');

    readln(inp);

    if inp = 'I' then
       info
    else if inp = 'D' then
       begin
          initialise;       {Read in starting values for demonstration program}
          run := ask_input (one_way);
       end
    else if inp = 'Q' then
       halt;

end;

END.
```

```
{$B+}    {Boolean complete evaluation on}
{$S+}    {Stack checking on}
{$I+}    {I/O checking on}
{$N+}    {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT GCINFO;        {Version 1      Created on 7th May 1991}

{$R+}    {Compiler range checking for arrays, etc.}
{$F+}    {Far call compiler directive}
{$O+}    {Overlay compiler directive}


{
 INFO is an overlaid unit of GCAP03, a Generalised Circuit Analysis Program
 for pulsed power.  This is version 1 of the unit which provides on-screen
 information on GCAP
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Printer;

procedure infy;
procedure geni;
procedure capb;
procedure stru;
procedure form;
procedure solu;
procedure acce;
procedure nonl;
procedure timv;
procedure user;
procedure libr;
procedure inp1;
procedure inp2;
procedure inp3;
procedure mory;
procedure info;

Implementation

{-------------------------------------------------------------------------}

{Now declare the procedures and functions}

{GCAP information options}
```

```
PROCEDURE INFY;

{
 INFY represents information on GCAP.  Procedure to display all available
 information options
}

begin

      clrscr;
      gotoxy(3,2);
      writeln('GCAP INFORMATION');
      writeln('----------------');

      geni;
      capb;
      stru;
      form;
      solu;
      acce;
      nonl;
      timv;
      user;
      libr;
      inp1;
      inp2;
      inp3;
      mory;

end;

PROCEDURE GENI;

{
 GENI represents General Information on GCAP.  Procedure to display GCAP
 general information, divided into subsections
}

begin

      clrscr;
      gotoxy(1,2);
      writeln('       GCAP GENERAL INFORMATION OPTIONS');
      writeln('       -------------------------------');
      writeln('     Information on GCAP is available in a menu format.');
      writeln('     Sub-menu topics are:');
      writeln('     1.  Complete information on GCAP');
      writeln('     2.  General information on GCAP');
      writeln('     3.  GCAP capabilities');
      writeln('     4.  Program structure');
      writeln('     5.  State equation formulation');
      writeln('     6.  Method of state equation solution');
      writeln('     7.  Acceptable circuit components');
      writeln('     8.  Nonlinear component modelling');
      writeln('     9.  Time-variant component modelling');
```

```
      writeln('     10. User-defined component modelling');
      writeln('     11. Library of GCAP-defined models');
      writeln('     12. Input file formatting : Component');
      writeln('     13. Input file formatting : Analysis');
      writeln('     14. Input file formatting : Structure');
      writeln('     15. Further information on GCAP');
      writeln('     16. Proceed with simulation');

end;

PROCEDURE CAPB;

{
 CAPB represents Capabilities.  Procedure to display GCAP capabilities
}

begin
      clrscr;
      gotoxy(1,2);
      writeln('          GCAP CAPABILITIES');
      writeln('          ----------------');
      writeln('            GCAP is a generalised circuit analysis program designed');
      writeln('          to simulate and analyse any network.  The size and');
      writeln('          complexity of the circuit is limited only by the available');
      writeln('          computer memory. GCAP may perform linear and nonlinear,');
      writeln('          time-invariant and time-varying DC, AC and transient');
      writeln('          analyses.  Circuits may contain linear/nonlinear,');
      writeln('          time-invariant/time-varying resistors, capacitors,');
      writeln('          inductors, coupled mutual inductors, independent voltage');
      writeln('          and current sources, switches, four types of dependent');
      writeln('          (V-V, V-I, I-V and I-I) voltage and current sources.  In');
      writeln('          addition, more complex components such as semiconductor');
      writeln('          devices (FET, BJT, diode, etc.) or transmission lines can');
      writeln('          be treated as a sub-circuit and defined either by the');
      writeln('          user or from the GCAP library of circuit models.  In either');
      writeln('          case, the user needs only to specify the pertinent model');
      writeln('          parameter values (in the former case, these will be');
      writeln('          user-defined).  Nonlinear and time-varying element models');
      writeln('          can be constructed either graphically or polynomially.');
      writeln('          Any collection of components may be defined as a subcircuit.');

end;

PROCEDURE STRU;

{
 STRU represents Structure.  Procedure to display GCAP program structure
}

begin

      clrscr;
      gotoxy(1,2);
      writeln('          GCAP PROGRAM STRUCTURE');
      writeln('          ---------------------');
```

```
    writeln('       GCAP is modular in structure, where each module can be');
    writeln('       refined in the initial development or extended by the user');
    writeln('       as necessary.  The user-defined function capability');
    writeln('       permits the inclusion of models of circuit components.');
    writeln('       Topological data describing circuit connections are input');
    writeln('       to GCAP in the form of component descriptors, which are');
    writeln('       then converted to a form acceptable to the program.');
    writeln('       Analysis descriptors determine the form of the analysis');
    writeln('       to be performed on the circuit. This may be either a DC,');
    writeln('       AC, transient or frequency analysis. Finally, the form of');
    writeln('       the output data is determined by structure descriptors to');
    writeln('       be in either tabular or graphics (two- or');
    writeln('       three-dimensional) form.');

end;

PROCEDURE FORM;

{
 FORM represents Formulation.  Procedure to display GCAP state equation
 formulation
}

begin

    clrscr;
    gotoxy(1,2);
    writeln('       STATE EQUATION FORMULATION');
    writeln('       ------------------------');
    writeln('         The basis of GCAP is the method of state-variable analysis');
    writeln('         (SVA).  The state equations are formulated as a minimal');
    writeln('         set of n first order differential equations describing the');
    writeln('         time behaviour of the state variables (capacitor voltages');
    writeln('         and inductor currents). These are, in general, non-linear');
    writeln('         functions with time-varying elements.  Topological data');
    writeln('         describing circuit connections is manipulated to obtain');
    writeln('         the state matrices A to E which can be substituted into');
    writeln('         the standard-form differential state and');
    writeln('         linear input-state-output equations.');

end;

PROCEDURE SOLU;

{
 SOLU represents Solution.  Procedure to display GCAP method of solution
}

begin

    clrscr;
    gotoxy(1,2);
    writeln('       STATE EQUATION SOLUTION');
    writeln('       ----------------------');
    writeln('         Numerical solution of the differential state equations in the');
```

```
    writeln('       time domain is by means of a fourth-order Runge-Kutta');
    writeln('       algorithm.  The algebraic solution of the linear state');
    writeln(,       equations is straightforward. By updating the state');
    writeln('       matrices at each successive time step, non-linear and');
    writeln('       time-varying characteristics are accounted for.');
    writeln('       The time step used in the algorithm is user-set. The');
    writeln('       execution time of GCAP is reduced by automatically');
    writeln('       increasing the time-step to one-fifth of the smallest');
    writeln('       time constant inherent in the A matrix.');

end;

PROCEDURE ACCE;

{
 ACCE represents Acceptable components.  Procedure to display GCAP
 acceptable circuit components
}

begin

    clrscr;
    gotoxy(1,2);
    writeln('       ACCEPTABLE CIRCUIT COMPONENTS');
    writeln('       ----------------------------');
    writeln('         GCAP can accomodate the following circuit components:');
    writeln('                                                              ');
    writeln('           Passive components: Resistors');
    writeln('                               Capacitors');
    writeln('                               Inductors');
    writeln('                               Independent/dependent');
    writeln('                                voltage and current sources');
    writeln('                                                              ');
    writeln('           GCAP Library:       Diodes');
    writeln('                               Bipolar junction transistors');
    writeln('                               Power MOSFETs');
    writeln('                               Transmission lines');
    writeln('                                                              ');
    writeln('           User-defined:       Switches');
    writeln('                               Time-varying sources');
    writeln('                               Flux-controlled magnetics');
    writeln('                               Voltage-controlled gas discharge');
    writeln('                                devices');

end;

PROCEDURE NONL;

{
 NONL represents Nonlinear.  Procedure to display GCAP nonlinear modelling
}

begin

    clrscr;
```

```
    gotoxy(1,2);
    writeln('       NONLINEAR COMPONENT MODELLING');
    writeln('       ----------------------------');
    writeln('         Non-linear dependence may be described either analytically');
    writeln('         or in piece-wise linear form. Examples of the former are');
    writeln('         the thyratron and saturable magnetic components. The state');
    writeln('         variables used to describe the non-linear elements are:');
    writeln('       ');
    writeln('            ELEMENT              CONTROL VARIABLE');
    writeln('                                 (STATE VARIABLE)');
    writeln('       ');
    writeln('            Capacitor            Voltage');
    writeln('                                 Charge');
    writeln('       ');
    writeln('            Magnetics            Current');
    writeln('                                 Flux ');
    writeln('       ');
    writeln('            Switches             Voltage');
    writeln('                                 Current');

end;

PROCEDURE TIMV;

{
 TIMV represents Time-varying.  Procedure to display GCAP time-varying
 modelling
}

begin

    clrscr;
    gotoxy(1,2);
    writeln('       TIME-VARYING COMPONENT MODELLING');
    writeln('       -------------------------------');
    writeln('         The time-dependent behaviour of components is entered to');
    writeln('         GCAP. Opening and closing times of switches are divided');
    writeln('         into chosen intervals. New state matrices are formulated');
    writeln('         at each successive time step. For continuity, the final');
    writeln('         switch states at any one time are the initial states at the');
    writeln('         next time step.');

end;

PROCEDURE USER;

{
 USER represents User-defined.  Procedure to display GCAP user-defined
 models
}

begin

    clrscr;
    gotoxy(1,2);
```

```
    writeln('       USER-DEFINED MODELLING');
    writeln('       ----------------------');
    writeln('         The modular structure of GCAP enebles the user to define');
    writeln('         new circuit elements. The model describing these elements');
    writeln('         are either:');
    writeln('        ');
    writeln('                              (1) Graphic');
    writeln('                              (2) Numerical');
    writeln('                              (3) Piece-wise linear');
    writeln('                              (4) Analytic');

end;

PROCEDURE LIBR;

{
 LIBR represents Library.  Procedure to display library of GCAP-defined
 models
}

begin

    clrscr;
    gotoxy(1,2);
    writeln('       GCAP-DEFINED MODEL LIBRARY');
    writeln('       -------------------------');
    writeln('         GCAP contains a library of built-in models of the most');
    writeln('         commonly-encountered circuit elements:');
    writeln('        ');
    writeln('                              (1) Diodes');
    writeln('                              (2) Bipolar junction transistors');
    writeln('                              (3) Power MOSFETs');
    writeln('                              (4) Transmission lines');

end;

PROCEDURE INP1;

{
 INP1 represents Input-file 1.  Procedure to display GCAP input file COMPON
 structure
}

begin

    clrscr;
    gotoxy(1,2);
    writeln('       INPUT FILE FORMATTING : COMPONENT');
    writeln('       --------------------------------');
    writeln('         Data specifying element type and value, nodal connections');
    writeln('         and any relevant non-linear/time-varying behaviour are');
    writeln('         entered in the form of component descriptors at the');
    writeln('         beginning of the program. Information is input on two');
    writeln('         separate lines, the first containing character data (to');
    writeln('         identify the element and its type) and the second');
```

```
    writeln('       numerical data (element value, nodal connections and any');
    writeln('       non-linear/time-varying parameters). Numbers to designate');
    writeln('       nodes must be positive integers but need not be');
    writeln('       consecutive. An algorithm is used to arrange this');
    writeln('       information into a form acceptable to GCAP.');
    writeln('       ');
    writeln('       The general form of a component descriptor is:');
    writeln('       ');
    writeln('         R     R******* (LIN/GRA/POL/USE) (VAR/INV)');
    writeln('         Value NANB     (Limit1) (Limit2) (Limit3) (Limit4)');
    writeln('                        (Limit5) (Limit6) (Limit7)');

end;

PROCEDURE INP2;

{
 INP2 represents Input-file 2.  Procedure to display GCAP input file ANALYS
 structure
}

begin

    clrscr;
    gotoxy(1,2);
    writeln('       INPUT FILE FORMATTING : ANALYSIS');
    writeln('       ------------------------------');
    writeln('         Analysis options are specified to GCAP in the form of');
    writeln('         analysis descriptors. Data is input on two separate lines,');
    writeln('         the first containing character data (to identify the');
    writeln('         analysis to be performed) and the second numerical data')
    writeln('         (time orfrequency range over which the simulation is to');
    writeln('         be run).');
    writeln('       ');
    writeln('         Analysis options may be expressed in terms of four');
    writeln('         categories:');
    writeln('       ');
    writeln('                              (1) Non-linear DC');
    writeln('                              (2) Linear AC');
    writeln('                              (3) Non-linear transient');
    writeln('                              (4) Frequency');
    writeln('       ');
    writeln('         Each analysis contains further sub-divisions.');
    writeln('       ');
    writeln('         T = Time interval of interest');
    writeln('         DT = Time increment (should be short compared with');
    writeln('         the time constants of the system');

end;

PROCEDURE INP3;

{
 INP3 represents Input-file 3.  Procedure to display GCAP input file STRUCT
 structure
```

```
}

begin

     clrscr;
     gotoxy(1,2);
     writeln('       INPUT FILE FORMATTING : STRUCTURE');
     writeln('       ---------------------------------');
     writeln('         Structure descriptors are general statements describing');
     writeln('         the formatting of the I/O data. These include the PRINT');
     writeln('         and PLOT commands, which are required for tabular');
     writeln('         listings of data and graphics, respectively. The user can');
     writeln('         specify which parameters are to be displayed. The voltage');
     writeln('         across, current through or power dissipated/energy stored');
     writeln('         in any circuit element can be monitored and output in the');
     writeln('         form of a printout over a user-specified time interval.');
     writeln('         Alternatively, a graphics subprogram provides a visual');
     writeln('         display of the results in either two- or three-dimensional');
     writeln('         form. Any additional output data such as comments');
     writeln('         headings, etc, are classed as structure descriptors.');
     writeln('        ');
     writeln('         Title may consist of up to fifty alphanumeric characters.');

end;

PROCEDURE MORY;

{
 MORY represents More information on GCAP.  Procedure to re-display
 GCAP information options
}

begin

     clrscr;
     gotoxy(1,2);
     writeln('       MORE INFORMATION ON GCAP');
     writeln('       -----------------------');
     writeln('       Further information on GCAP is available in the User Manual');

end;

PROCEDURE INFO;

{
 INFO represents information on GCAP.  Procedure to display the available
 GCAP information options
}

var   cha, ip          :integer;

begin              {Start of code section for INFO}
     clrscr;
     gotoxy(1,2);
        writeln('INFORMATION ON GCAP');
```

```
    writeln('====================');
    writeln('1.   Complete information on GCAP');
    writeln('2.   General information on GCAP');
    writeln('3.   GCAP capabilities');
    writeln('4.   Program structure');
    writeln('5.   State equation formulation');
    writeln('6.   Method of state equation solution');
    writeln('7.   Acceptable circuit components');
    writeln('8.   Nonlinear component modelling');
    writeln('9.   Time-variant component modelling');
    writeln('10.  User-defined component modelling');
    writeln('11.  Library of GCAP-defined models');
    writeln('12.  Input file formatting : Component');
    writeln('13.  Input file formatting : Analysis');
    writeln('14.  Input file formatting : Structure');
    writeln('15.  Further information on GCAP');
    writeln('16.  Proceed with simulation');
    writeln;
    writeln('Enter no. of sub-topic of interest');

readln(cha);
if cha = 1 then
    infy
else if cha = 2 then
    geni
else if cha = 3 then
    capb
else if cha = 4 then
    stru
else if cha = 5 then
    form
else if cha = 6 then
    solu
else if cha = 7 then
    acce
else if cha = 8 then
    nonl
else if cha = 9 then
    timv
else if cha = 10 then
    user
else if cha = 11 then
    libr
else if cha = 12 then
    inp1
else if cha = 13 then
    inp2
else if cha = 14 then
    inp3
else if cha = 15 then
    mory
else if cha = 16 then
    exit;
    writeln;
    clreol;
```

```
        writeln('Press any number to return to main menu');
        writeln(' or press 16 to proceed with simulation');
        readln(ip);
        if ip <> 16 then info;

end;

END.
```

```
{$B+}   {Boolean complete evaluation on}
{$S+}   {Stack checking on}
{$I+}   {I/O checking on}
{$N+}   {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT GCLIBR;        {Version 1        Created on 7th May 1991}

{$R+}   {Compiler range checking for arrays, etc.}
{$F+}   {Far call compiler directive}
{$O+}   {Overlay compiler directive}


{
 LIBR is an overlaid unit of GCAP03, a Generalised Circuit Analysis Program
 for pulsed power.  This is version 1 of the unit which contains a library
 of complex circuit element models
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Printer,
  GcInit;

procedure model;

Implementation

{-------------------------------------------------------------------------}

{Now declare the procedures and functions}

PROCEDURE MODEL;

VAR number,
      d2, d3                    :integer;

      d1, d4                    :real;

procedure diode;

{
 Procedure to model the diode
}

begin

     clrscr;
```

```
      gotoxy(1,2);
      writeln('        DIODE MODELLING');
      writeln('        ---------------');
      writeln('        The GCAP model of a diode is based on the diode junction');
      writeln('        law characteristic');

         for number := 1 to num1 do
         begin
            if COMP7[number,1] = 'D' then
            begin

               d1 := COMP8[number,1];
               d2 := COMP8[number,2];
               d3 := COMP8[number,3];
               d4 := COMP8[number,4]
*(((VOLTAGE[number,round(time/inc_time)]*e)/(k*temperature)) - 1);

               COMP7[number,1] := 'R';
               COMP7[number,2] := 'Rdiode';
               COMP7[number,3] := 'NON';
               COMP7[number,4] := 'INV';
               COMP7[number,5] := 'Limit1';
               COMP7[number,6] := 'Limit2';
               COMP7[number,7] := 'Limit3';
               COMP8[number,1] := d1;
               COMP8[number,2] := d2;
               COMP8[number,3] := d3;
               COMP8[number,4] := d4;

            end;
         end;
end;

procedure bipolar;

{
 Procedure to model the bipolar junction transistor
}

begin

      clrscr;
      gotoxy(1,2);
      writeln('        BIPOLAR JUNCTION TRANSISTOR MODELLING');
      writeln('        ------------------------------------');
      writeln('        The GCAP model of a bipolar junction transistor is based');
      writeln('        on the Ebers-Moll characteristic');

         for number := 1 to num1 do
         begin
            if COMP7[number,1] = 'BJ' then
            begin

               d1 := COMP8[number,1];
               d2 := COMP8[number,2];
```

```
                d3 := 101;
                d4 := COMP8[number,4]
*(((VOLTAGE[number,round(time/inc_time)]*e)/(k*temperature)) - 1);

                COMP7[number,1] := 'D';
                COMP7[number,2] := 'D1';
                COMP7[number,3] := 'NON';
                COMP7[number,4] := 'INV';
                COMP7[number,5] := 'Limit1';
                COMP7[number,6] := 'Limit2';
                COMP7[number,7] := 'Limit3';
                COMP8[number,1] := d1;
                COMP8[number,2] := d2;
                COMP8[number,3] := d3;
                COMP8[number,4] := d4;

                d1 := COMP8[number,1];
                d2 := 101;
                d3 := COMP8[number,3];
                d4 := COMP8[number,4]
*(((VOLTAGE[number,round(time/inc_time)]*e)/(k*temperature)) - 1);

                COMP7[number,1] := 'D';
                COMP7[number,2] := 'D2';
                COMP7[number,3] := 'NON';
                COMP7[number,4] := 'INV';
                COMP7[number,5] := 'Limit1';
                COMP7[number,6] := 'Limit2';
                COMP7[number,7] := 'Limit3';
                COMP8[number,1] := d1;
                COMP8[number,2] := d2;
                COMP8[number,3] := d3;
                COMP8[number,4] := d4;

                d1 := COMP8[number,1];
                d2 := COMP8[number,2];
                d3 := 101;
                d4 := COMP8[number,4]*CURRENT[number,round(time/inc_time)];

                COMP7[number,1] := 'I';
                COMP7[number,2] := 'I1';
                COMP7[number,3] := 'LIN';
                COMP7[number,4] := 'INV';
                COMP7[number,5] := 'Limit1';
                COMP7[number,6] := 'Limit2';
                COMP7[number,7] := 'Limit3';
                COMP8[number,1] := d1;
                COMP8[number,2] := d2;
                COMP8[number,3] := d3;
                COMP8[number,4] := d4;

                d1 := COMP8[number,1];
                d2 := COMP8[number,2];
                d3 := 101;
                d4 := COMP8[number,4]*CURRENT[number,round(time/inc_time];
```

```
                COMP7[number,1] := 'I';
                COMP7[number,2] := 'I2';
                COMP7[number,3] := 'LIN';
                COMP7[number,4] := 'INV';
                COMP7[number,5] := 'Limit1';
                COMP7[number,6] := 'Limit2';
                COMP7[number,7] := 'Limit3';
                COMP8[number,1] := d1;
                COMP8[number,2] := d2;
                COMP8[number,3] := d3;
                COMP8[number,4] := d4;

            end;
        end;
end;

procedure mosfet;

{
 Procedure to model the power MOSFET
}

begin

    clrscr;
    gotoxy(1,2);
    writeln('       POWER MOSFET MODELLING');
    writeln('       ---------------------');
    writeln('       The GCAP model of a power MOSFET is based on the');
    writeln('       Schichmann-Hodges characteristic');

        for number := 1 to num1 do
        begin
            if COMP7[number,1] = 'MF' then
            begin

                d1 := COMP8[number,1];
                d2 := COMP8[number,2];
                d3 := 121;
                d4 := COMP8[number,4]
                      *0.5*SQR(VOLTAGE[number,round(time/inc_time)]);

                COMP7[number,1] := 'I';
                COMP7[number,2] := 'I1';
                COMP7[number,3] := 'LIN';
                COMP7[number,4] := 'INV';
                COMP7[number,5] := 'Limit1';
                COMP7[number,6] := 'Limit2';
                COMP7[number,7] := 'Limit3';
                COMP8[number,1] := d1;
                COMP8[number,2] := d2;
                COMP8[number,3] := d3;
                COMP8[number,4] := d4;
```

```
            end;
        end;
end;

procedure transmission_line;

{
 Procedure to model the transmission line
}

begin

     clrscr;
     gotoxy(1,2);
     writeln('       TRANSMISSION LINE MODELLING');
     writeln('       -------------------------');
     writeln('       The GCAP model of a transmission line consists of N');
     writeln('       pi-networks');

end;

procedure thyratron;

{
 Procedure to model the thyratron
}

begin

     clrscr;
     gotoxy(1,2);
     writeln('       THYRATRON MODELLING');
     writeln('       ------------------');
     writeln('       The GCAP model of a thyratron is analytical');

end;

PROCEDURE spark_gap;

{
 Procedure to model the spark gap
}

begin

     clrscr;
     gotoxy(1,2);
     writeln('       SPARK GAP MODELLING');
     writeln('       -----------------');
     writeln('       The GCAP model of a spark gap is piece-wise linear');

end;

begin          {Start of code section for MODEL}
```

```
    for num := 1 to num1 do
    begin
       if COMP7[num,1] = 'D' then
       begin
          diode;
       end
       else if COMP7[num,1] = 'BJ' then
       begin
          bipolar;
       end
       else if COMP7[num,1] = 'MF' then
       begin
          mosfet;
       end;
    end;
end;

END.
```

```
{$B+}   {Boolean complete evaluation on}
{$S+}   {Stack checking on}
{$I+}   {I/O checking on}
{$N+}   {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT GCUSER;        {Version 1      Created on 7th May 1991}

{$R+}   {Compiler range checking for arrays, etc.}
{$F+}   {Far call compiler directive}
{$O+}   {Overlay compiler directive}


{
 LIBR is an overlaid unit of GCAP03, a Generalised Circuit Analysis Program
 for pulsed power.  This is version 1 of the unit which is used to interface
 to a user-defined model.
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Printer,
  GcInit;

procedure user_model;

Implementation

{-------------------------------------------------------------------------}

{Now declare the procedures and functions}

PROCEDURE USER_MODEL;

VAR num, number                 :integer;

      Y1, Y2, Y3                :string[6];

      A1, A2, A3,
      B1, B2, B3,
      X1, X2, X3          :real;

procedure dependence;

begin

     {Functional dependence of user_model}
     {Dependent parameters are B1, B2, B3, Y1, Y2, Y3}
```

```
end;

begin          {Start of code section for USER_MODEL}

    for num := 1 to num1 do
    begin
      if (COMP7[num,1] = 'SUB1') or (COMP7[num,1] = 'SUB2')
       or (COMP7[num,1] = 'SUB3') then
      begin
        A1 := 0;
        A2 := 0;
        A3 := 0;
        X1 := 0;
        X2 := 0;
        X3 := 0;

        dependence;
        writeln('Enter explicit dependence of sub_model resistance:');
        writeln('Dependent parameter no.1:');
          readln('Y1');
        writeln('Explicit dependence:');
          readln(B1);
          for number = 1 to num1 do
          begin
            if COMP7[number,1] = 'Y1' then
            begin
              A1 := B1;
              X1 := COMP8[number,1];
            end;
          end;

        writeln('Dependent parameter no.2:');
          readln('Y2');
        writeln('Explicit dependence:');
          readln(B2);
          for number = 1 to num1 do
          begin
            if COMP7[number,1] = 'Y2' then
            begin
              A2 := B2;
              X2 := COMP8[number,1];
            end;
          end;

        writeln('Dependent parameter no.3:');
          readln('Y3');
        writeln('Explicit dependence:');
          readln(B3);
          for number = 1 to num1 do
          begin
            if COMP7[number,1] = 'Y3' then
            begin
              A3 := B3;
              X3 := COMP8[number,1];
```

```
            end;
          end;

        COMP8[num,1] := (A1*X1) + (A2*X2) + (A3*X3);

      end;
    end;

END.
```

```
{$B+}   {Boolean complete evaluation on}
{$S+}   {Stack checking on}
{$I+}   {I/O checking on}
{$N+}   {Numeric coprocessor}
{$M 65500,16384,655360}{Turbo 3 default stack and heap}


UNIT GCINPT;        {Version 1    Created on 11th June 1991}

{$R+}   {Compiler range checking for arrays, etc.}
{$F+}   {Far call compiler directive}
{$O+}   {Overlay compiler directive}


{
 GCINPT is an overlaid unit of GCAP03, a Generalised Circuit Analysis Program
 for pulsed power.  This is version 1 of the unit which is used to
 input data to the program
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Printer,
  GcInit,
  GcFile;

{procedure comp;
procedure anly;
procedure strc; }
procedure enter;

Implementation

{-------------------------------------------------------------------------}

{Now declare the procedures and functions}


PROCEDURE ENTER;

{
 INPUT FILES: User-input data to construct three files:-

      (1) A complete description of the topology of the circuit to be
          analysed : data is stored in the two-dimensional array "COMPON"

      (2) A specification of the analyses to be performed : stored in the
          two-dimensional array "ANALYS"
```

```
and     (3) General structural information relevant to the program output
            formatting : stored in the two-dimensional array "STRUCT"

}

var   descrip1                  :array[0..7] of string[6];
      descrip2                  :array[1..4] of real;
      ch                        :real;

      num, ip1, ip2, nom        :integer;

      arrayc, arraya, arrays    :array[1..20] of string[6];

PROCEDURE COMP;

{
 COMP represents Component.  Procedure to store all circuit component and
 topology data in arrays "COMP1" and "COMP2" - common "COMPON".  Circuit
 component types, values and nodal connections, together with any relevant
 information concerning nonlinearity/time-variance, whether a group of
 elements is a sub-circuit/GCAP- or user-defined model, etc., are input
 according to the format description of the circuit.  Component data is
 stored in the two-dimensional array "COMPON"

 Variables used are:-

     N1 :  Parameter used to read up to max_num_comp data items into arrays
           "COMP1" and "COMP2"
     N2 :  Parameter used to read up to four character data items into
           array "COMP1"
     N3 :  Parameter used to read up to eighteen real numerical data items
           into array "COMP2"

 max_num_comp :  Maximum number of circuit elements (component descriptors)
}

var   N, M              :integer;

begin

     num1 := num1 + 1;
     for N := 1 to 7 do
        COMP1[num1,N] := descrip1[N];
     for M := 1 to 4 do
        COMP2[num1,M] := descrip2[M];

end;

PROCEDURE ANLY;

{
 ANLY represents Analysis.  Procedure to store all analysis specification
 data in arrays "ANLY1" and "ANLY2" - common "ANALYS"

 Variables used are:-
```

```
    N4 :  Parameter used to read up to max_num_anly data items into arrays
          "ANLY1" and "ANLY2"
    N5 :  Parameter used to read up to four character data items into
          array "ANLY1"
    N6 :  Parameter used to read up to eight real numerical data items
          into array "ANLY2"

 max_num_anly :  Maximum number of analysis descriptors
}

var   N, M            :integer;

begin

     num2 := num2 + 1;
     for N := 1 to 7 do
        ANLY1[num2,N] := descrip1[N];
     for M := 1 to 4 do
        ANLY2[num2,M] := descrip2[M];

end;

PROCEDURE STRC;

{
 STRC represents Structure.  Procedure to store all GCAP output formatting
 data in arrays "STRC1" and "STRC2" - common "STRUCT"

 Variables used are:-

     N7 :  Parameter used to read up to max_num_strc data items into arrays
           "STRC1" and "STRC2"
     N8 :  Parameter used to read up to four character data items into
           array "STRC1"
     N9 :  Parameter used to read up to eight realnumerical data items
           into array "STRC2"

 max_num_strc :  Maximum number of structure descriptors
}

var   N, M            :integer;

begin

     num3 := num3 + 1;
     for N := 1 to 7 do
        STRC1[num3,N] := descrip1[N];
     for M := 1 to 4 do
        STRC2[num3,M] := descrip2[M];

end;


begin     {Start of code section for ENTER}
```

```
num := 0;
num1 := 0;
num2 := 0;
num3 := 0;

arrayc[1] := 'R';
arrayc[2] := 'C';
arrayc[3] := 'L';
arrayc[4] := 'ML';
arrayc[5] := 'V';
arrayc[6] := 'I';
arrayc[7] := 'VI';
arrayc[8] := 'VV';
arrayc[9] := 'II';
arrayc[10] := 'IV';
arrayc[11] := 'D';
arrayc[12] := 'BJ';
arrayc[13] := 'JF';
arrayc[14] := 'MF';
arrayc[15] := 'SR';
arrayc[16] := 'AR';
arrayc[17] := 'GT';
arrayc[18] := 'OP';
arrayc[19] := 'SU';
arrayc[20] := 'NA';

arraya[1] := 'TIME';
arraya[2] := 'RESPON';
arraya[3] := 'STABLE';
arraya[4] := 'SENSE';
arraya[5] := 'TEMPRE';
arraya[6] := 'DCTRAN';
arraya[7] := 'POLESH';
arraya[8] := 'POLES';
arraya[9] := 'BODE';
arraya[10] := 'IMPULS';
arraya[11] := 'ZEROS';
arraya[12] := 'GAINM';
arraya[13] := 'PHASEM';
arraya[14] := 'FBRESP';
arraya[15] := 'STABM';
arraya[16] := 'FILES';
arraya[17] := 'ACTRAN';
arraya[18] := 'NA';
arraya[19] := 'NA';
arraya[20] := 'NA';

arrays[1] := 'PRIMAT';
arrays[2] := 'INFO';
arrays[3] := 'TITLE';
arrays[4] := 'PRINT';
arrays[5] := 'PLOT2';
arrays[6] := 'PLOT3';
arrays[7] := 'NA';
```

```
arrays[8] := 'NA';
arrays[9] := 'NA';
arrays[10] := 'NA';
arrays[11] := 'NA';
arrays[12] := 'NA';
arrays[13] := 'NA';
arrays[14] := 'NA';
arrays[15] := 'NA';
arrays[16] := 'NA';
arrays[17] := 'NA';
arrays[18] := 'NA';
arrays[19] := 'NA';
arrays[20] := 'NA';

clrscr;
gotoxy(1,1);
writeln('Data is input to GCAP in the form of descriptors');
writeln;
writeln('       CIRCUIT DESCRIPTION');
writeln('       ------------------');
writeln('       Enter data to describe the circuit in accordance with the');
writeln('       format described in section 10 of GCAP Information');
writeln;
writeln('       ANALYSIS DESCRIPTION');
writeln('       -------------------');
writeln('       Enter data to specify the types of analysis to be performed');
writeln('       on the circuit in accordance with the format described in');
writeln('       section 11 of GCAP Information');
writeln;
writeln('       STRUCTURE DESCRIPTION');
writeln('       --------------------');
writeln('       Enter data to describe the form of the output from GCAP in');
writeln('       accordance with the format described in section 12 of GCAP');
writeln('       Information.  Information may be displayed by typing "INFO"');
writeln;
writeln('       Type E to complete data');
writeln;
writeln('Hit any number to continue');
readln(ch);

repeat

clrscr;
gotoxy(1,1);
   writeln(sp5,'Enter a descriptor symbol:');
gotoxy(1,3);
writeln(sp10,'1.  Component                              2. Analysis    3. Structure');
writeln(sp10,'    ---------                              --------       ---------');
writeln(sp10,'    R : Resistor                           TIME           PRIMAT');
writeln(sp10,'    C : Capacitor                          RESPON         INFO');
writeln(sp10,'    L : Inductor                           STABLE         TITLE');
writeln(sp10,'    ML:Mutual inductor                     SENSE          PRINT');
writeln(sp10,'    V : Independent voltage source         TEMPRE         PLOT2');
writeln(sp10,'    I :  Independent current source        DCTRAN         PLOT3');
writeln(sp10,'    VI: Voltage-controlled current source  POLESH');
```

```
writeln(sp10,'    VV:Voltage-controlled voltage source    POLES');
writeln(sp10,'    II:  Current-controlled current source  BODE');
writeln(sp10,'    IV: Current-controlled voltage source   IMPULS');
writeln(sp10,'    D : Diode                               ZEROS');
writeln(sp10,'    BJ: BJT                                 GAINM');
writeln(sp10,'    JF: JFET                                PHASEM');
writeln(sp10,'    MF:MOSFET                               FBRESP');
writeln(sp10,'    SR: SCR                                 STABM');
writeln(sp10,'    AR:ASCR                                 FILES');
writeln(sp10,'    GT:GTO                                  ACTRAN');
writeln(sp10,'    OP: OP AMP');
writeln(sp10,'    SU: Sub-circuit');

   gotoxy(1,24);
      writeln(sp18,'S : Start,  E : End of descriptors,  Q : Quit GCAP');

         writeln;
         readln(descrip1[0]);
         writeln('Option  ');
         readln(descrip1[1]);

         if (descrip1[0] = 'Q') or (descrip1[1] = 'Q') then
            halt
         else if descrip1[1] = 'E' then
            files
         else
            begin
            for nom := 1 to 20 do
               begin
               if (descrip1[1] = arrayc[nom]) then
                  begin
                  num := num + 1;

                  writeln('Component symbol  ');
                  readln(descrip1[2]);
                  writeln('Nonlinear(NON)/Linear(LIN)  ');
                  readln(descrip1[3]);
                  writeln('Time-varying(VAR)/Time-invariant(INV)  ');
                  readln(descrip1[4]);
                  writeln('Limit 1  ');
                  readln(descrip1[5]);
                  writeln('Limit 2  ');
                  readln(descrip1[6]);
                  writeln('Limit 3  ');
                  readln(descrip1[7]);

                  writeln('Component value  ');
                  read(descrip2[1]);
                  writeln('Nodal connection number A  ');
                  read(descrip2[2]);
                  writeln('Nodal connection number B  ');
                  read(descrip2[3]);
                  writeln('Nonlinear break-point  ');
                  read(descrip2[4]);
```

```
    comp;
    end
else if (descrip1[1] = arraya[nom]) then
    begin
    num := num + 1;

    writeln('N/A   ');
    readln(descrip1[2]);
    writeln('N/A   ');
    readln(descrip1[3]);
    writeln('N/A   ');
    readln(descrip1[4]);
    writeln('Limit 1   ');
    readln(descrip1[5]);
    writeln('Limit 2   ');
    readln(descrip1[6]);
    writeln('Limit 3   ');
    readln(descrip1[7]);

    writeln('Increment time   ');
    read(descrip2[1]);
    writeln('Simulation time   ');
    read(descrip2[2]);
    writeln('N/A   ');
    read(descrip2[3]);
    writeln('N/A   ');
    read(descrip2[4]);

    anly;
    end

else if (descrip1[1] = arrays[nom]) then
    begin
    num := num + 1;

    writeln('Format   ');
    readln(descrip1[2]);
    writeln('Parameter   ');
    readln(descrip1[3]);
    writeln('N/A   ');
    readln(descrip1[4]);
    writeln('Limit 1   ');
    readln(descrip1[5]);
    writeln('Limit 2   ');
    readln(descrip1[6]);
    writeln('Limit 3   ');
    readln(descrip1[7]);

    writeln('Output increment   ');
    read(descrip2[1]);
    writeln('Output range   ');
    read(descrip2[2]);
    writeln('N/A   ');
    read(descrip2[3]);
    writeln('N/A   ');
```

```
                    read(descrip2[4]);

                    strc;
                    end;
                end;
{               else
                    writeln('Illegal data item : Input again');    }
            end;

    until descrip1[1] = 'E';

end;        {End of code section for ENTER}

END.
```

```
{$B+}   {Boolean complete evaluation on}
{$S+}   {Stack checking on}
{$I+}   {I/O checking on}
{$N+}   {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT GCINIT;            {Version 1      Created on 7th May 1991}
                        {Last modified on 12th June 1991}

{$R+}   {Compiler range checking for arrays, etc.}
{$F+}   {Far call compiler directive}
{$O+}   {Overlay compiler directive}


{
 INIT is an overlaid unit of GCAP03, a Generalised Circuit Analysis Program
 for pulsed power.  This is version 1 of the unit which is used to initialise
 the input parameters of the circuit employed in the demonstration program.
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Printer;


TYPE  text_index =  'A'..'~';

CONST cr :char = ^m; lf :char = ^j; ff :char = ^l; bel :char = ^g;
      sp1 = ' '; sp2 = '  ';
      sp3 = '   '; sp4 = '    ';
      sp5 = '     '; sp6 = '      ';
      sp7 = '       '; sp8 = '        ';
      sp9 = '         '; sp10 = '          ';
      sp11 = '           '; sp12 = '            ';
      sp13 = '             '; sp14 = '              ';
      sp15 = '               '; sp16 = '                ';
      sp17 = '                 '; sp18 = '                  ';
      sp19 = '                   '; sp20 = '                    ';
      sp21 = '                     '; sp22 = '                      ';
      sp23 = '                       '; sp24 = '                        ';
      sp25 = '                         ';
      sp26 = '                          ';
      sp27 = '                           ';
      sp28 = '                            ';
      sp29 = '                             ';
      sp30 = '                              ';
      sp31 = '                               ';
      sp32 = '                                ';
```

```
sp33 = '                                        ';
sp34 = '                                        ';
sp35 = '                                        ';
sp36 = '                                        ';

{Declare the fundamental constants of the system}

pi  = 3.14;                 {pi}
vcl = 3E8;                  {Velocity of light in m/s}
k   = 1.38E-23;             {Boltzmann constant}
e   = 1.602E-19;            {Electronic charge}
h   = 6.63E-34;             {Planck's constant}

{Declare the general constants}

ME = 9.109E-31;             {Electron mass}

max_num_comp = 20;  {Maximum number of components in circuit}
max_num_anly = 4;   {Maximum number of analysis options}
max_num_strc = 5;   {Maximum number of structure options}

{   max_num_W = 3; }   {Maximum number of tree branch independent
                        voltage sources in circuit}
{   max_num_DD = 3; }  {Maximum number of tree branch capacitances in
                        circuit}
{   max_num_BB = 3; }  {Maximum number of tree branch conductances in
                        circuit}
{   max_num_EE = 3; }  {Maximum number of tree branch reciprocal
                        inductances in circuit}
{   max_num_U = 3;  }  {Maximum number of link elastances in circuit}
{   max_num_X = 3;  }  {Maximum number of link resistances in circuit}
{   max_num_V = 3;  }  {Maximum number of link inductances in circuit}
{   max_num_AA = 3; }  {Maximum number of link independent current
                        sources in circuit}

max_num_AX = 10;    {Maximum dimension of the A-matrix
                     No. of passive components in circuit}
max_num_BX = 10;    {Maximum dimension of the B-matrix}
max_num_CX = 10;    {Maximum dimension of the C-matrix}
max_num_DX = 10;    {Maximum dimension of the D-matrix}
max_num_EX = 10;    {Maximum dimension of the E-matrix}

max_num_points = 100; {Maximum number of simulation points}

max_num_branch = 5;  {Maximum number of tree branches in circuit}
max_num_link = 10;   {Maximum number of tree branches and links
                      in circuit}

VAR  line    :   array [text_index] of string [55];

     {ANALYS}    edit_rev, save_data     :char;
                 graph_table             :string[5];
                 upper_A, upper_B,
                 lower_A, lower_B        :integer;
```

```
                    sim_time, inc_time,
                    upper_range_H, upper_range_L,
                    lower_range_H, lower_range_L   :real;

    {ELECTR}        volt_supply,
                    cap_stor, cap_peak,
                    ind_charge, ind_unsat,
                    rel_perm, sat_curr,
                    res_bypass, ind_bypass,
                    res_loop, ind_loop             :real;

    {LASTHY}        res_laser, ind_laser,
                    res_thyr_off, res_thyr_on,
                    ind_thyr, thyr_on_time         :real;

    title                                          :string[40];

    run, iterate, one_way, firstime                :boolean;


    time                                           :real;

    num1, num2, num3                               :integer;

    T1,
    W, DD, BB, EE, U, X, V, AA                     :integer;

    COMP1       :array[1..max_num_comp,1..7] of string[6];
    COMP2       :array[1..max_num_comp,1..4] of real;
    ANLY1       :array[1..max_num_anly,1..7] of string[6];
    ANLY2       :array[1..max_num_anly,1..4] of real;
    STRC1       :array[1..max_num_strc,1..7] of string[6];
    STRC2       :array[1..max_num_strc,1..4] of real;

    COMP7       :array[1..max_num_comp,1..7] of string[6];
    COMP8       :array[1..max_num_comp,1..4] of real;

    VOLTAGE     :array[1..max_num_AX,0..max_num_points] of real;
    CURRENT     :array[1..max_num_AX,0..max_num_points] of real;

CONST LIMIT1= 3;      {Maximum dimensions (rows) of matrices}
      LIMIT2= 3;      {Maximum dimensions (columns) of matrices}

TYPE  VECTOR= array [1..LIMIT1,1..LIMIT2] of real;
      SCALAR= array [1..LIMIT1,1..LIMIT2] of integer;

VAR   PROD, PLUS, MINU,
      TRAN, INVS, IDEN,
      ZERO, MAT13, MAT14        :vector;

procedure initialise;
function ask_input(var one_way :boolean):boolean;

Implementation
```

```
{-------------------------------------------------------------------------}

{Now declare the procedures and functions}


PROCEDURE INITIALISE;

begin     {First set up the array of text strings}

{ ANALYSIS LIMITS }
    line ['A']  := 'EDIT/REVIEW PARAMETERS (Y/N):';
    line ['B']  := 'MAXIMUM SIMULATION TIME (ns):';
    line ['C']  := 'TIME INCREMENT (ps):';
    line ['D']  := 'UPPER TRACE A:';
    line ['E']  := 'UPPER TRACE B:';
    line ['F']  := 'UPPER TRACE RANGE (HIGH/LOW):';
    line ['G']  := 'LOWER TRACE A:';
    line ['H']  := 'LOWER TRACE B:';
    line ['I']  := 'LOWER TRACE RANGE (HIGH/LOW):';
    line ['J']  := 'CRT PLOT (GRAPH) OR TABLE (TABLE):';
    line ['K']  := 'SAVE DATA TO USER FILE (Y/N):';

{ (B) ELECTRICAL STATE VARIABLES (ELECTR) }
    line ['g']  := 'POWER SUPPLY VOLTAGE (kV):';
    line ['h']  := 'STORAGE CAPACITANCE (nF): ';
    line ['i']  := 'PEAKING CAPACITANCE (nF):';
    line ['j']  := 'CHARGING INDUCTANCE (mH):';
    line ['k']  := 'UNSATURATED MPC INDUCTANCE (microhenries):';
    line ['l']  := 'RELATIVE PERMEABILITY OF FERRITE:';
    line ['m']  := 'SATURATION CURRENT (amps):';
    line ['n']  := 'BYPASS ELEMENT RESISTANCE (ohms): ';
    line ['o']  := 'BYPASS ELEMENT INDUCTANCE (microhenries): ';
    line ['p']  := 'DISCHARGE LOOP RESISTANCE (ohms): ';
    line ['q']  := 'DISCHARGE LOOP INDUCTANCE (nanohenries): ';

{ (C) LASER AND THYRATRON STATE VARIABLES (LASTHY) }
    line ['t']  := 'LASER HEAD RESISTANCE (ohms):';
    line ['u']  := 'LASER HEAD INDUCTANCE (nH):';
    line ['v']  := 'THYRATRON OFF-RESISTANCE (ohms): ';
    line ['w']  := 'ARC-DROP THYRATRON RESISTANCE (ohms): ';
    line ['x']  := 'THYRATRON INDUCTANCE (nH): ';
    line ['y']  := 'THYRATRON TURN-ON TIME (ns): ';

    {Now initialise the main parameters to default values}

        edit_rev := 'N';
        sim_time := 3010.0;
        inc_time := 20.0;
        upper_A := 1;
        upper_B := 2;
        upper_range_H := 0.0;
        upper_range_L := 0.0;
        lower_A := 3;
        lower_B := 4;
        lower_range_H := 0.0;
```

```
lower_range_L := 0.0;
graph_table := 'TABLE';
save_data := 'N';

volt_supply := 7.2;
cap_stor := 4.0;
cap_peak := 3.0;
ind_charge := 150.0;
res_laser := 1.0;
ind_laser := 500.0;
res_thyr_off := 100000.0;
res_thyr_on := 2.0;
ind_thyr := 500.0;
thyr_on_time := 5.0;
res_bypass := 5.0;
ind_bypass := 100.0;
res_loop := 1.0;
ind_loop := 1.0;
ind_unsat := 100.0;
sat_curr := 75.0;
rel_perm := 500;

COMP7[1,1] := 'V';
COMP7[1,2] := 'Vsource';
COMP7[1,3] := 'LIN';
COMP7[1,4] := 'INV';
COMP7[1,5] := 'N/A';
COMP7[1,6] := 'N/A';
COMP7[1,7] := 'N/A';
COMP8[1,1] := 14.4E3;
COMP8[1,2] := 1;
COMP8[1,3] := 2;
COMP8[1,4] := 1E6;

COMP7[2,1] := 'L';
COMP7[2,2] := 'Lcharge';
COMP7[2,3] := 'LIN';
COMP7[2,4] := 'INV';
COMP7[2,5] := 'N/A';
COMP7[2,6] := 'N/A';
COMP7[2,7] := 'N/A';
COMP8[2,1] := 150E-3;
COMP8[2,2] := 2;
COMP8[2,3] := 3;
COMP8[2,4] := 1E6;

COMP7[3,1] := 'C';
COMP7[3,2] := 'Cstor';
COMP7[3,3] := 'LIN';
COMP7[3,4] := 'INV';
COMP7[3,5] := 'N/A';
COMP7[3,6] := 'N/A';
COMP7[3,7] := 'N/A';
COMP8[3,1] := 4.0E-9;
COMP8[3,2] := 3;
```

```
COMP8[3,3] := 9;
COMP8[3,4] := 1E6;

COMP7[4,1] := 'R';
COMP7[4,2] := 'Rthyr';
COMP7[4,3] := 'NON';
COMP7[4,4] := 'INV';
COMP7[4,5] := 'N/A';
COMP7[4,6] := 'N/A';
COMP7[4,7] := 'N/A';
COMP8[4,1] := 1.0E5;
COMP8[4,2] := 3;
COMP8[4,3] := 4;
COMP8[4,4] := 2.0;

COMP7[5,1] := 'L';
COMP7[5,2] := 'Lthyr';
COMP7[5,3] := 'LIN';
COMP7[5,4] := 'INV';
COMP7[5,5] := 'N/A';
COMP7[5,6] := 'N/A';
COMP7[5,7] := 'N/A';
COMP8[5,1] := 500E-9;
COMP8[5,2] := 4;
COMP8[5,3] := 1;
COMP8[5,4] := 1E6;

COMP7[6,1] := 'R';
COMP7[6,2] := 'Rloop';
COMP7[6,3] := 'LIN';
COMP7[6,4] := 'INV';
COMP7[6,5] := 'N/A';
COMP7[6,6] := 'N/A';
COMP7[6,7] := 'N/A';
COMP8[6,1] := 1.0;
COMP8[6,2] := 5;
COMP8[6,3] := 6;
COMP8[6,4] := 1E6;

COMP7[7,1] := 'L';
COMP7[7,2] := 'Lloop';
COMP7[7,3] := 'LIN';
COMP7[7,4] := 'INV';
COMP7[7,5] := 'N/A';
COMP7[7,6] := 'N/A';
COMP7[7,7] := 'N/A';
COMP8[7,1] := 1.0E-9;
COMP8[7,2] := 1;
COMP8[7,3] := 5;
COMP8[7,4] := 1E6;

COMP7[8,1] := 'R';
COMP7[8,2] := 'Rlaser';
COMP7[8,3] := 'NON';
COMP7[8,4] := 'INV';
```

```
            COMP7[8,5] := 'N/A';
            COMP7[8,6] := 'N/A';
            COMP7[8,7] := 'N/A';
            COMP8[8,1] := 1.0E4;
            COMP8[8,2] := 9;
            COMP8[8,3] := 7;
            COMP8[8,4] := 10.0;

            COMP7[9,1] := 'L';
            COMP7[9,2] := 'Llaser';
            COMP7[9,3] := 'NON';
            COMP7[9,4] := 'INV';
            COMP7[9,5] := 'N/A';
            COMP7[9,6] := 'N/A';
            COMP7[9,7] := 'N/A';
            COMP8[9,1] := 500E-9;
            COMP8[9,2] := 7;
            COMP8[9,3] := 6;
            COMP8[9,4] := 100E-9;

            COMP7[10,1] := 'R';
            COMP7[10,2] := 'Rbypass';
            COMP7[10,3] := 'LIN';
            COMP7[10,4] := 'INV';
            COMP7[10,5] := 'N/A';
            COMP7[10,6] := 'N/A';
            COMP7[10,7] := 'N/A';
            COMP8[10,1] := 5.0;
            COMP8[10,2] := 8;
            COMP8[10,3] := 6;
            COMP8[10,4] := 1E6;

            COMP7[11,1] := 'L';
            COMP7[11,2] := 'Lbypass';
            COMP7[11,3] := 'LIN';
            COMP7[11,4] := 'INV';
            COMP7[11,5] := 'N/A';
            COMP7[11,6] := 'N/A';
            COMP7[11,7] := 'N/A';
            COMP8[11,1] := 100E-6;
            COMP8[11,2] := 8;
            COMP8[11,3] := 6;
            COMP8[11,4] := 1E6;


        firstime := false;
        one_way := false;
end;

FUNCTION  ask_input (var one_way :boolean):boolean;

{
 Input of data to GCAP03.  Contains a nested procedure param designed to act
 like a screen editor.
}
```

```
var       index                                          :text_index;
          key                                            :char;


procedure param (which :text_index; edit :boolean);

{
 Parameter input procedure designed to behave like a screen editor. The
 calling parameter WHICH specifies which of the model parameters is involved,
 and EDIT specifies whether it is to be displayed or edited. Each parameter
 is input at a distinct place on the screen, and the cursor is moved there to
 allow the new value to be entered after its prompt. The prompt doubles as a
 descriptor of the quantity for display purposes.
}

var       row :integer;

begin     {Start of code section for PARAM. First work out where to put the
           prompt on the screen}

     if which in ['A'..'K']
       then row := (ORD (which) - ORD ('A'))
     else if which in ['g'..'q']
       then row := (ORD (which) - ORD ('g'))
     else if which in ['t'..'y']
       then row := (ORD (which) - ORD ('t'));
     row := row + 6;
     gotoxy (1,row);
     clreol;                  {Erase previous value of the prompt and value}
      if (which in ['A'..'K']) or (which in ['g'..'q']) or
         (which in ['t'..'y'])
      then write (sp10, which + '....' + line [which]  );       {Put up new prompt}

      case which of
           'A'  : if edit
                    then read (edit_rev)
                    else write (edit_rev:30);
           'B'  : if edit
                    then read (sim_time)
                    else write (sim_time:30:2);
           'C'  : if edit
                    then read (inc_time)
                    else write (inc_time:39:2);
           'D'  : if edit
                    then read (upper_A)
                    else write (upper_A:45);
           'E'  : if edit
                    then read (upper_B)
                    else write (upper_B:45);
           'F'  : if edit
                    then read (upper_range_H, upper_range_L)
                    else write (upper_range_H:15:2, upper_range_L:15:2);
           'G'  : if edit
                    then read (lower_A)
```

```
        else write (lower_A:45);
'H'  : if edit
        then read (lower_B)
        else write (lower_B:45);
'I'  : if edit
        then read (lower_range_H, lower_range_L)
        else write (lower_range_H:15:2, lower_range_L:15:2);
'J'  : if edit
        then read (graph_table)
        else write (graph_table:25);
'K'  : if edit
        then read (save_data)
        else write (save_data:30);
'g'  : if edit
        then read (volt_supply)
        else write (sp27, volt_supply:6:2);
'h'  : if edit
        then read (cap_stor)
        else write (sp27, cap_stor:6:2);
'i'  : if edit
        then read (cap_peak)
        else write (sp28, cap_peak:6:2);
'j'  : if edit
        then read (ind_charge)
        else write (sp28, ind_charge:6:2);
'k'  : if edit
        then read (ind_unsat)
        else write (sp11, ind_unsat:6:2);
'l'  : if edit
        then read (rel_perm)
        else write (sp19, rel_perm:6:2);
'm'  : if edit
        then read (sat_curr)
        else write (sp27, sat_curr:6:2);
'n'  : if edit
        then read (res_bypass)
        else write (sp20, res_bypass:5:1);
'o'  : if edit
        then read (ind_bypass)
        else write (sp12, ind_bypass:5:1);
'p'  : if edit
        then read (res_loop)
        else write (sp20, res_loop:5:1);
'q'  : if edit
        then read (ind_loop)
        else write (sp14, ind_loop:4:1);
't'  : if edit
        then read (res_laser)
        else write (sp26,res_laser:6:2);
'u'  : if edit
        then read (ind_laser)
        else write (sp26,ind_laser:6:2);
'v'  : if edit
        then read (res_thyr_off)
        else write (sp17,res_thyr_off:6:2);
```

```
            'w'  : if edit
                     then read (res_thyr_on)
                     else write (sp15,res_thyr_on:6:2);
            'x'  : if edit
                     then read (ind_thyr)
                     else write (sp26,ind_thyr:6:2);
            'y'  : if edit
                     then read (thyr_on_time)
                     else write (sp24,thyr_on_time:6:2);

      end;  {Case statement on which}

end;        {End of code section for PARAM}


procedure edit_prompt1;

{
 Nested procedure called from ASK_INPUT relating to the editing of analysis
 limits
}

begin       {Start of code section for EDIT_PROMPT1}

      gotoxy (1,24);
      clreol;
      write ('Hit menu to edit a value, 1 : edit a parameter, 2 : run, 3 : quit');

end;        {End of code section for EDIT_PROMPT1}


procedure edit_prompt2;

{
 Nested procedure called from ASK_INPUT relating to the editing of operating
 parameters
}

begin       {Start of code section for EDIT_PROMPT2}

      gotoxy (1,24);
      clreol;
      write ('Hit menu to edit a value, 1 : view next page, 2 : run, 3 : quit');

end;        {End of code section for EDIT_PROMPT2}


procedure laserthyr;

var   index                              :text_index;

begin
          clrscr;
          gotoxy (1,3);
          writeln (sp4,title,': LASER AND THYRATRON PARAMETERS',cr,lf);
```

```
          for index := 't' to 'y' do
              param (index, firstime);
          repeat
                for index := 't' to 'y' do
                    param (index, false);
                gotoxy (1,24);
                clreol;
                write ('Hit menu to edit a value, 2 : run, or 3 : quit');
                Key := Readkey;
{! 5. USE TURBO3 ^unit for access to KBD, or instead use CRT and ReadKey}
                if key in ['t'..'y']
                  then param (key, true);
                if key in ['3']
                  then halt;
                if key in ['2']
                  then ask_input := true;
          until key in ['2'];

end;   {Of ASK_INPUT function}

procedure electr;

var   index                                :text_index;

begin
          clrscr;
          gotoxy (1,3);
          writeln (sp4, title, ': ELECTRICAL PARAMETERS', cr, lf);
          for index := 'g' to 'q' do
              param (index, firstime);
     repeat
                for index := 'g' to 'q' do
                    param (index, false);
                edit_prompt2;
                Key := Readkey;
{! 4. USE TURBO3 ^unit for access to KBD, or instead use CRT and ReadKey}
                if key in ['g'..'q']
                  then param (key, true);
                if key in ['3']
                  then halt;
                if key in ['2']
                  then ask_input := true;
                if key in ['1'] then
                   begin
                      laserthyr;
                   end;
     until key in ['2'];
end;


begin       {Start of code section for ASK_INPUT}

     ask_input := false;
     clrscr;
     writeln('==============================================',lf);
```

```
    writeln('                    G. C. A. P.                    ',lf);
    writeln('COMPUTER MODELLING OF ELECTRICAL CIRCUITS',lf);
    writeln('       BY STATE-VARIABLE ANALYSIS',lf);
    writeln('=============================================',lf);
    writeln;
    writeln('GCAP is a Generalised Circuit Analysis Program');
    writeln('designed to simulate electrical circuit behaviour',lf);
    writeln ('Title of simulation: ');
    title := 'GCAP Analysis : Noname';
    readln (title);

    clrscr;
    gotoxy (1,3);
    writeln (sp10, title ,': ANALYSIS LIMITS', cr, lf);
    for index := 'A' to 'K' do
        param (index, firstime);
    repeat
        for index := 'A' to 'K' do
        param (index, false);
        edit_prompt1;
        Key := Readkey;
{! 1. USE TURBO3 ^unit for access to KBD, or instead use CRT and ReadKey}
        if key in ['A'..'K']
          then param (key, true);
        if key in ['3']
          then halt;
        if key in ['2']
          then ask_input := true;
        if key in ['1'] then electr;

    until key in ['2'];

        firstime := false;      {So next time round we show the values at once}

end;          {Of ASK_INPUT function}

END.
```

```
{$B+}   {Boolean complete evaluation on}
{$S+}   {Stack checking on}
{$I+}   {I/O checking on}
{$N+}   {Numeric coprocessor}
{$M 65500,16384,655360}{Turbo 3 default stack and heap}


UNIT GCFILE;        {Version 1   Created on 7th May 1991}

{$R+}   {Compiler range checking for arrays, etc.}
{$F+}   {Far call compiler directive}
{$O+}   {Overlay compiler directive}


{
 FILE is an overlaid unit of GCAP03, a Generalised Circuit Analysis Program
 for pulsed power.  This is version 1 of the unit which is used to store all
 circuit element, analysis and format descriptor information in files
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Printer,
  GcInit;


{Declare the fundamental constants of the system}

procedure files;

Implementation

{-------------------------------------------------------------------------}

{Now declare the procedures and functions}

PROCEDURE FILES;

{
 FILES is a procedure to count, arrange into files and display the following:
                    (a) number and type of each element in the circuit
                    (b) analyses to be performed
                and (c) format of output data
}

var   num4, num5, num6, num7,
      num8, num9, num10, num11,
      num12, num13, num14, num15          :integer;
```

```
      ch                                        :real;

begin
     clrscr;
     gotoxy(1,4);
     writeln('Input files COMPONENT, ANALYSIS and STRUCTURE');
     writeln(' are complete:');
     writeln('              ');

     writeln('COMPONENT DESCRIPTORS:');
     writeln('              ');
     num7 := 0;
     for num4 := 1 to num1 do
     begin
        num7 := num7 + 1;
        gotoxy(1,4+(3*num7));
        for num5 := 1 to 7 do
        write(COMP1[num4,num5]:11);
        writeln;
        for num6 := 1 to 4 do
        write('              ',COMP2[num4,num6]:11);
     end;
     writeln('              ');

     writeln('ANALYSIS DESCRIPTORS:');
     writeln('              ');
     num11 := 0;
     for num8 := 1 to num2 do
     begin
        num11 := num11 + 1;
        gotoxy(1,4+(3*num11));
        for num9 := 1 to 7 do
        write(ANLY1[num8,num9]:11);
        writeln;
        for num10 := 1 to 4 do
        write('              ',ANLY2[num8,num10]:11);
     end;
     writeln('              ');

     writeln('STRUCTURE DESCRIPTORS:');
     writeln('              ');
     num15 := 0;
     for num12 := 1 to num3 do
     begin
        num15 := num15 + 1;
        gotoxy(1,4+(3*num15));
        for num13 := 1 to 7 do
        write(STRC1[num12,num13]:11);
        writeln;
        for num14 := 1 to 4 do
        write('              ',STRC2[num12,num14]:11);
     end;
     writeln('              ');

     gotoxy(1,24);
```

```
    writeln('Hit any number to continue');
    readln(ch);

end;    {Of code section of files}

END.
```

```
{$B+}   {Boolean complete evaluation on}
{$S+}   {Stack checking on}
{$I+}   {I/O checking on}
{$N+}   {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT GCSORT;        {Version 1      Created on 11th June 1991}

{$R+}   {Compiler range checking for arrays, etc.}
{$F+}   {Far call compiler directive}
{$O+}   {Overlay compiler directive}


{
 GCSORT is an overlaid unit of GCAP03, a Generalised Circuit Analysis Program
 for pulsed power.  This is version 1 of the unit which is used to
 sort the network branches into a proper tree
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Printer,
  GcInit;

{procedure sort2;
procedure find;
procedure noddup;
procedure order;
procedure elim;
procedure nodcov;
procedure nodcon;
procedure dimens;
procedure consis;
procedure connec;     }
procedure sort;

Implementation

{-------------------------------------------------------------------------}

{Now declare the procedures and functions}

PROCEDURE SORT;

{
 Sorts circuit branches into a proper tree
}
```

```
CONST     max_num_nodes = 20;  {Maximum number of nodes in circuit}

var   maxnod, minnod,
      M1,
      NB, MX                  :integer;

      NODES                   :array[0..max_num_nodes] of integer;

      COMP3                   :array[1..max_num_comp,1..7] of string[6];
      COMP4                   :array[1..max_num_comp,1..4] of real;

      element                 :string[6];


PROCEDURE SORT2;

var   NX, NY                  :integer;

begin

{
 Scan COMP2(N,2) and COMP2(N,3) and identify maximum node number maxnod
 }

      maxnod := 0;

      for NX := 1 to num1 do
      begin
         if (COMP2[NX,2] > maxnod) then
            maxnod := round(COMP2[NX,2]);
         if (COMP2[NX,3] > maxnod) then
            maxnod := round(COMP2[NX,3]);
         writeln(lst,'num1 = ',num1);
         writeln(lst,'COMP2[',NX,',2] = ',COMP2[NX,2]:11);
         writeln(lst,'COMP2[',NX,',3] = ',COMP2[NX,3]:11);
      end;
      writeln(lst,'maxnod = ',maxnod);

{
 Scan COMP2(N,2) and COMP2(N,3) and identify all node numbers
 }

      MX := 0;
      NODES[0] := 0;

      for NY := 1 to maxnod do
      begin
         for NX := 1 to num1 do
         begin
            if (round(COMP2[NX,2]) = NY) then
            begin
               if (round(COMP2[NX,2]) > NODES[MX]) then
               begin
                  MX := MX + 1;
                  NODES[MX] := round(COMP2[NX,2]);
```

```
                writeln(lst,'NX = ',NX,'  NODES[',MX,'] = ',NODES[MX]);
            end;
        end;
        if (COMP2[NX,3] = NY) then
        begin
            if (round(COMP2[NX,3]) > NODES[MX]) then
            begin
                MX := MX + 1;
                NODES[MX] := round(COMP2[NX,3]);
                writeln(lst,'NX = ',NX,'  NODES[',MX,'] = ',NODES[MX]);
            end;
        end;
    end;
end;

{
Identify minimum node number minnod
}

    minnod := NODES[1];

{
Calculate number of nodes
}

    clrscr;
    M1 := MX;
    T1 := M1 - 1;
    writeln(lst,'        Number of nodes          :   ',MX);
    writeln;
    writeln(lst,'        Number of branches       :   ',num1);
    writeln;
    writeln(lst,'        Number of tree branches  :   ',T1);
    writeln(lst,'maxnod = ',maxnod);
    writeln(lst,'minnod = ',minnod);

end;

PROCEDURE FIND(element: string);

{
Scan COMP1(N,1) for an independent voltage source (V), a capacitance (C),
a resistance (R), an inductance (L) or an independent current source (I).
If found, store in new arrays COMP3(M,1) and COMP4(M,1)
}

var   NA, NC, ND                                :integer;

begin
    for NA := 1 to num1 do
    begin
        if (COMP1[NA,1] = element) then
        begin
            NB := NB + 1;
            for NC := 1 to 7 do
```

```
            begin
               COMP3[NB,NC] := COMP1[NA,NC];
               writeln(lst,'COMP3[',NB,',',NC,'] = ',COMP3[NB,NC]);
            end;
            for ND := 1 to 4 do
            begin
               COMP4[NB,ND] := COMP2[NA,ND];
               writeln(lst,'COMP4[',NB,',',ND,'] = ',COMP4[NB,ND]);
            end;
         end;
      end;
end;

PROCEDURE NODDUP(element: char);

{
 Procedure to ensure that no two nodes are surplus to requirements via a
 particular element type.  If they are, then remove from new arrays COMP7
 and COMP8
}

begin

      writeln(lst,'NODDUP');              {Superceded by SORT2}

end;

PROCEDURE ORDER(element: char);

{
 Procedure to arrange the nodes into ascending order
}

begin

      writeln(lst,'ORDER');               {Superceded by SORT2}

end;

PROCEDURE ELIM;

var   a, b, a1, a2, a3,
      a4, a5, a6                          :integer;

      param1, param2                      :boolean;

begin

      for a := 1 to 7 do
         COMP7[1,a] := COMP3[1,a];

      writeln(lst,'COMP7[1,1] = ',COMP7[1,1]);

      for b := 1 to 4 do
         COMP8[1,b] := COMP4[1,b];
```

```
a1 := 1;
a2 := 0;
a4 := 0;

for a3 := 2 to num1 do
begin
   a4 := a4 + 1;

   param1 := true;
   param2 := true;

   for a5 := 1 to a4 do
   begin
      if ((COMP4[a3,2]=COMP8[a5,2]) or (COMP4[a3,2]=COMP8[a5,3])) then
      begin
         param1 := false;
      end;
      if ((COMP4[a3,3]=COMP8[a5,2]) or (COMP4[a3,3]=COMP8[a5,3])) then
      begin
         param2 := false;
      end;
   end;

      if ((not param1) and (not param2)) then
      begin

            a2 := a2 + 1;

            for a := 1 to 7 do
            begin
               COMP7[T1+a2,a] := COMP3[a3,a];
            end;

            if (COMP3[a3,1] = 'C') then
               COMP7[T1+a2,1] := 'S';

            for b := 1 to 4 do
            begin
               COMP8[T1+a2,b] := COMP4[a3,b];
            end;
      end
      else
      begin
         a1 := a1 + 1;

            for a := 1 to 7 do
            begin
               COMP7[a1,a] := COMP3[a3,a];
               writeln(lst,'COMP7[',a1,',',a,'] = ',COMP7[a1,a]);
            end;

            if (COMP3[a3,1] = 'R') then
               COMP7[a1,1] := 'G'
            else if (COMP3[a3,1] = 'L') then
```

```
                    COMP7[a1,1] := 'T';

                for b := 1 to 4 do
                begin
                    COMP8[a1,b] := COMP4[a3,b];
                    writeln(lst,'COMP8[',a1,',',b,'] = ',COMP8[a1,b]);
                end;
            end;

        writeln(lst,'COMP7[',a3,',1] = ',COMP7[a3,1]);

        end;

        for a := 1 to num1 do
        begin
            writeln(lst,'COMP7[',a,',1] = ',COMP7[a,1]);
            writeln(lst,'COMP7[',a,',2] = ',COMP7[a,2]);
        end;
end;

PROCEDURE NODCOV;

var   a1                                    :integer;

begin

        for a1 := 1 to MX do
        begin
            if ((COMP8[a1,2]=NODES[a1]) or (COMP8[a1,3]=NODES[a1])) then
                begin
                    writeln(lst,'Node no. ',a1, ' is covered');
                    { COMP9 }
            end;
        end;

{
 Are all nodes covered?
 }

end;

PROCEDURE NODCON;

var   a1                                    :integer;

begin

        for a1 := 1 to MX do
        begin
            if ((COMP8[a1,2]=NODES[a1]) or (COMP8[a1,3]=NODES[a1])) then
                begin
                    writeln(lst,'Node no. ',a1, ' is connected');
                    { COMP10 }
            end;
        end;
```

```
{
 Are all nodes connected?
 }

end;

PROCEDURE DIMENS;

var   NH                        :integer;

begin

      W := 0;
      DD := 0;
      BB := 0;
      EE := 0;
      U := 0;
      X := 0;
      V := 0;
      AA := 0;

      for NH := 1 to num1 do
      begin
         if (COMP7[NH,1] = 'V') then
            W := NH
         else if (COMP7[NH,1] = 'C') then
            DD := NH - W
         else if (COMP7[NH,1] = 'G') then
            BB := NH - (W+DD)
         else if (COMP7[NH,1] = 'T') then
            EE := NH - (W+DD+BB)
         else if (COMP7[NH,1] = 'S') then
            U := NH - (W+DD+BB+EE)
         else if (COMP7[NH,1] = 'R') then
            X := NH - (W+DD+BB+EE+U)
         else if (COMP7[NH,1] = 'L') then
            V := NH - (W+DD+BB+EE+U+X)
         else if (COMP7[NH,1] = 'I') then
            AA := NH - (W+DD+BB+EE+U+X+V);
      end;

      writeln(lst,'W = ',W);
      writeln(lst,'DD = ',DD);
      writeln(lst,'BB = ',BB);
      writeln(lst,'EE = ',EE);
      writeln(lst,'U = ',U);
      writeln(lst,'X = ',X);
      writeln(lst,'V = ',V);
      writeln(lst,'AA = ',AA);

end;

PROCEDURE CONSIS;
```

```
{
 Consistency check to compare the total number of circuit elements entered
 by the user with the circuit description entered
}

var    WX, DDX, BBX, EEX, UX, XX, VX, AAX       :integer;

begin

   repeat
     clrscr;
     gotoxy(1,2);
     writeln(sp6,'CONSISTENCY CHECK');
     writeln(sp6,'----------------');
     writeln(sp6,'          ');
     writeln(sp6,'NUMBER OF INDEPENDENT VOLTAGE SOURCES');
     read(WX);
     writeln(sp6,'NUMBER OF CAPACITANCES');
     read(DDX);
     writeln(sp6,'NUMBER OF RESISTANCES');
     read(XX);
     writeln(sp6,'NUMBER OF INDUCTANCES');
     read(VX);
     writeln(sp6,'NUMBER OF INDEPENDENT CURRENT SOURCES');
     read(AAX);

     if (WX <> W) then
        gotoxy(4,1);
        write('Error in number of independent voltage sources, input again');
     if (DDX <> DD+U) then
        gotoxy(4,1);
        write('Error in number of capacitances, input again');
     if (XX <> BB+X) then
        gotoxy(4,1);
        write('Error in number of resistances, input again');
     if (VX <> EE+V) then
        gotoxy(4,1);
        write('Error in number of inductances, input again');
     if (AAX <> AA) then
        gotoxy(4,1);
        write('Error in number of independent current sources, input again');
        readln;
   until (WX+DDX+XX+VX+AAX = num1);


end;


PROCEDURE CONNEC;

{
 Procedure to Connect the branches via data from SORT for use by MATQ
 }

begin
```

```
    writeln(lst,'CONNEC');              {Superceded by SORT2}
end;
begin                   {Start of code section for SORT}
    sort2;
    NB := 0;
    find('V');          {Scan COMP1(N,1) for an independent voltage source (V).
                         If found, store in new arrays COMP3(M,1) and COMP4(M,1)}
    noddup('V');        {Check that no two nodes are surplus to requirements via
                         voltage sources. If they are, then remove from arrays
                         COMP7 and COMP8}
    order('V');
    find('C');          {Scan COMP1(N,1) for a capacitance (C). If found, store
                         in new arrays COMP3(M,1) and COMP4(M,1) }
    noddup('C');        {Check that no two nodes are surplus to requirements via
                         capacitances. If they are, then remove from arrays
                         COMP7 and COMP8}
    order('C');
    find('R');          {Scan COMP1(N,1) for a resistance (R). If found, store
                         in new arrays COMP3(M,1) and COMP4(M,1) }
    noddup('R');        {Check that no two nodes are surplus to requirements via
                         resistances. If they are, then remove from arrays
                         COMP7 and COMP8}
    order('R');
    find('L');          {Scan COMP1(N,1) for an inductance (L). If found, store
                         in new arrays COMP3(M,1) and COMP4(M,1) }
    noddup('L');        {Check that no two nodes are surplus to requirements via
                         inductances. If they are, then remove from arrays
                         COMP7 and COMP8}
    order('L');
    find('I');          {Scan COMP1(N,1) for an independent current source (I).
                         If found, store in new arrays COMP3(M,1) and COMP4(M,1)}
    noddup('I');        {Check that no two nodes are surplus to requirements via
                         current sources. If they are, then remove from arrays
                         COMP7 and COMP8}
    order('I');
    elim;
    nodcov;
    nodcon;
    dimens;
    consis;
```

```
        connec;
end;
END.
```

```
{$B+}   {Boolean complete evaluation on}
{$S+}   {Stack checking on}
{$I+}   {I/O checking on}
{$N+}   {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT GCGRAP;        {Version 1      Created on 11th June 1991}

{$R+}   {Compiler range checking for arrays, etc.}
{$F+}   {Far call compiler directive}
{$O+}   {Overlay compiler directive}

{
 GCGRAP is an overlaid unit of GCAP03, a Generalised Circuit Analysis Program
 for pulsed power.  This is version 1 of the unit which provides a graphics
 display of the circuit under simulation. The graphics output is
 two-dimensional.
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Graph,
  Printer,
  GcInit;

procedure graphics;

Implementation

{Start of global type, constant and variable definitions}

VAR    GraphDriver                                          :integer;
       GraphMode                                            :integer;
       ErrorCode                                            :integer;

       PixelColor, Size                                     :word;

{CONST Gray50:    FillPatternType =  ($AA, $55, $AA, $55,
                                     $AA, $55, $AA, $55);  }

{-------------------------------------------------------------------------}

{Now declare the procedures and functions}


PROCEDURE GRAPHICS;

{
```

```
 This procedure is called from SIMULATE each time-step to provide a
 two-dimensional graphics display of the circuit to be simulated
}

begin                    {Start of code section for GRAPHICS}

     writeln(lst,'GRAPHICS');

     GraphDriver := Detect;                          {Set flag: do detection}
     InitGraph(GraphDriver, GraphMode, 'C:\DRIVERS');
     ErrorCode := GraphResult;
     if ErrorCode <> grOk then                       {Is there an error?}
     begin
          writeln('Graphics error: ', GraphErrorMsg(ErrorCode));
          writeln('Program aborted...');
          Halt(1);
     end;

end;          {End of code section of Graphics}

END.
```

```
{$B+}   {Boolean complete evaluation on}
{$S+}   {Stack checking on}
{$I+}   {I/O checking on}
{$N+}   {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT GCMAT;        {Version 1    Created on 7th May 1991}

{$R+}   {Compiler range checking for arrays, etc.}
{$F+}   {Far call compiler directive}
{$O+}   {Overlay compiler directive}


{
 MAT is an overlaid unit of GCAP03, a Generalised Circuit Analysis Program
 for pulsed power.  This is version 1 of the unit which is used to
 manipulate matrix operation
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Printer,
  GcInit;

{Declare the general constants of the system}

procedure matmul(MAT1, MAT2:vector; MX, NX, PX, QX:integer);
procedure matplu(MAT4, MAT5:vector; MX, NX, PX, QX:integer);
procedure matmin(MAT6, MAT7:vector; MX, NX, PX, QX:integer);
procedure mattrn(MAT8:vector; MX, NX, PX, QX:integer);
procedure matinv(MAT9:vector; MX, NX, PX, QX:integer);
procedure matidn(MAT10:vector; MX, NX, PX, QX:integer);
procedure matzer(MAT11:vector; MX, NX, PX, QX:integer);
procedure matprd(MAT12:vector; CONS:real; MX, NX, PX, QX:integer);
procedure matmch(MAT15:vector; MX, NX, PX, QX:integer);
procedure matpri(MAT3:vector;ROWS1, COLS2:integer);

Implementation

{------------------------------------------------------------------------}

{Now declare the procedures and functions}

PROCEDURE MATMUL(MAT1, MAT2:vector; MX, NX, PX, QX:integer);
{
 Procedure to calculate the product of an M x N matrix with a P x Q matrix.
 N must equal P for the product to be defined (MATCH = TRUE), and the
 product PROD is then an M x Q matrix
```

Variables used are:-

```
        LIMIT1 :  Limit on dimensions (rows) of the matrices
        LIMIT2 :  Limit on dimensions (columns) of the matrices
        MX     :  Number of rows in first matrix
        NX     :  Number of columns in first matrix
        PX     :  Number of rows in second matrix
        QX     :  Number of columns in second matrix
        MAT1   :  The first matrix
        MAT2   :  The second matrix
        PROD   :  The product of MAT1 and MAT2
        IX     :  One of the matrix indices
        JX     :  One of the matrix indices
        KX     :  One of the matrix indices
        SUM    :  Used to calculate the product matrix
}

var   IX, JX, KX                    :integer;

      SUM                           :real;

      MATCH                         :boolean;

begin

      {if (NX = PX) then
      begin    }
         MATCH := true;
         for IX := 1 to MX do
         begin
            for JX := 1 to QX do
            begin
               SUM := 0.0;
               for KX := 1 to NX do
               begin
                  SUM := SUM + (MAT1[IX,KX]*MAT2[KX,JX]);
               end;
               PROD[IX,JX] := SUM;
            end;
         end;
      { end
       else
         MATCH := false; }

end;

PROCEDURE MATPLU(MAT4, MAT5:vector; MX, NX, PX, QX:integer);

{
 Procedure to calculate the addition of an M x N matrix with a second M x N
 matrix.  The matrices must both be of these dimensions for the addition to be
 defined (MATCH = TRUE), and the addition PLUS is then also an M x N
 matrix
```

Variables used are:-

```
LIMIT  :  Limit on dimensions of the matrices
MAT4   :  The first matrix
MAT5   :  The second matrix
PLUS   :  The addition of MAT5 and MAT6
MX     :  Number of rows in both matrices
QX     :  Number of columns in both matrices
IX     :  One of the matrix indices
JX     :  One of the matrix indices
```

}

```
var   IX, JX                    :integer;

      MATCH                     :boolean;

begin

      if (MX = PX) then
      begin
         MATCH := true;
         for IX := 1 to MX do
         begin
            for JX := 1 to QX do
            begin
              PLUS[IX,JX] := MAT4[IX,JX] + MAT5[IX,JX];
            end;
         end;
      end
      else
         MATCH := false;

end;
```

```
PROCEDURE MATMIN(MAT6, MAT7:vector; MX, NX, PX, QX:integer);
```

{
Procedure to calculate the subtraction of an M x N matrix from a second M x N matrix. The matrices must both be of these dimensions for the addition to be defined (MATCH = TRUE), and the subtraction MINU is then also an M x N matrix

Variables used are:-

```
LIMIT  :  Limit on dimensions of the matrices
MAT6   :  The first matrix
MAT7   :  The second matrix
MINU   :  The subtraction of MAT7 from MAT6
MX     :  Number of rows in both matrices
QX     :  Number of columns in both matrices
IX     :  One of the matrix indices
JX     :  One of the matrix indices
```

}

```
var   IX, JX                          :integer;

      MATCH                           :boolean;

begin

      if (NX = PX) then
      begin
         MATCH := true;
         for IX := 1 to MX do
         begin
            for JX := 1 to QX do
            begin
               MINU[IX,JX] := MAT6[IX,JX] - MAT7[IX,JX];
            end;
         end;
      end
      else
         MATCH := false;

end;


PROCEDURE MATTRN(MAT8:vector; MX, NX, PX, QX:integer);

{
 Procedure to calculate the transpose of an M x N matrix.  The transpose is
 an N x M matrix

 Variables used are:-

          LIMIT :  Limit on dimensions of the matrix
          MAT8  :  The matrix to be transposed
          TRAN  :  The resulting transposed matrix
          MX    :  Number of rows in MAT8 and columns in TRAN
          QX    :  Number of columns in MAT8 and rows in TRAN
          IX    :  One of the matrix indices
          JX    :  One of the matrix indices
}
var   IX, JX                          :integer;

      MATCH                           :boolean;

begin

      if (NX = PX) then
      begin
         MATCH := true;
         for IX := 1 to MX do
         begin
            for JX := 1 to QX do
            begin
               TRAN[IX,JX] := MAT8[JX,IX];
```

```
        end;
      end;
    end
    else
      MATCH := false;

end;


PROCEDURE MATINV(MAT9:vector; MX, NX, PX, QX:integer);

{
 Procedure to calculate the inverse of an M x N matrix.  The inverse is also
 an M x N matrix

 Variables used are:-

          LIMIT :  Limit on dimensions of the matrix
          MAT9  :  The matrix to be inverted
          INVS  :  The resulting inverted matrix
          MX    :  Number of rows in both matrices
          QX    :  Number of columns in both matrices
          IX    :  One of the matrix indices
          JX    :  One of the matrix indices
}

var   IX, JX                          :integer;

      MATCH                           :boolean;

begin

    if (NX = PX) then
    begin
      MATCH := true;
      for IX := 1 to MX do
      begin
        for JX := 1 to QX do
        begin
          if (MAT9[IX,JX] < 1E-15) then
            INVS[IX,JX] := 0
          else
            INVS[IX,JX] := 1/MAT9[IX,JX];
          writeln(lst,'INVS[',IX,',',JX,'] = ',INVS[IX,JX]:11);
        end;
      end;
    end
    else
      MATCH := false;

end;


PROCEDURE MATIDN(MAT10:vector; MX, NX, PX, QX:integer);
```

```
{
Procedure to equate an M x N matrix to the identity matrix.  The identity
matrix is also an M x N matrix

Variables used are:-

        LIMIT :  Limit on dimensions of the matrix
        MAT10 :  The matrix to be equated to the identity matrix
        IDEN  :  The resulting identity matrix
        MX    :  Number of rows in both matrices
        QX    :  Number of columns in both matrices
        IX    :  One of the matrix indices
        JX    :  One of the matrix indices
}

var   IX, JX                          :integer;

      MATCH                           :boolean;

begin

      if (NX = PX) then
      begin
         MATCH := true;
         for IX := 1 to MX do
         begin
            for JX := 1 to QX do
            begin
               IDEN[IX,JX] := 1;
            end;
         end;
      end
      else
         MATCH := false;

end;


PROCEDURE MATZER(MAT11:vector; MX, NX, PX, QX:integer);

{
Procedure to equate an M x N matrix to the zero matrix.  The zero matrix is
also an M x N matrix

Variables used are:-

        LIMIT :  Limit on dimensions of the matrix
        MAT11 :  The matrix to be equated to the zero matrix
        ZERO  :  The resulting zero matrix
        MX    :  Number of rows in both matrices
        QX    :  Number of columns in both matrices
        IX    :  One of the matrix indices
        JX    :  One of the matrix indices
}
```

```
var   IX, JX                          :integer;

      MATCH                           :boolean;

begin

      if (NX = PX) then
      begin
         MATCH := true;
         for IX := 1 to MX do
         begin
            for JX := 1 to QX do
            begin
               ZERO[IX,JX] := 0.0;
            end;
         end;
      end
      else
         MATCH := false;

end;


PROCEDURE MATPRD(MAT12:vector; CONS:real; MX, NX, PX, QX:integer);

{
 Procedure to calculate the product of a constant with an M x N matrix.
 The resulting matrix is also an M x N matrix

 Variables used are:-

          LIMIT :  Limit on dimensions of the matrix
          MAT12 :  The matrix to be multiplied by a constant
          MAT13 :  The resulting matrix
          CONST :  The multiplying constant
          MX    :  Number of rows in the matrix
          QX    :  Number of columns in the matrix
          IX    :  One of the matrix indices
          JX    :  One of the matrix indices
}
var   IX, JX                          :integer;

      MATCH                           :boolean;

begin

      if (NX = PX) then
      begin
         MATCH := true;
         for IX := 1 to MX do
         begin
            for JX := 1 to QX do
            begin
               MAT13[IX,JX] := CONS*MAT12[IX,JX];
```

```
            end;
         end;
      end
      else
         MATCH := false;

end;


PROCEDURE MATMCH(MAT15:vector; MX, NX, PX, QX:integer);

{
 Procedure to equate an M x N matrix to a second M x N matrix.  The resulting
 matrix is also an M x N matrix

 Variables used are:-

          LIMIT :  Limit on dimensions of the matrices
          MAT14 :  The matrix to be matched
          MAT15 :  The matrix to be matched to
          MX    :  Number of rows in both matrices
          QX    :  Number of columns in both matrices
          IX    :  One of the matrix indices
          JX    :  One of the matrix indices
}

var   IX, JX                          :integer;

      MATCH                           :boolean;

begin

      if (NX = PX) then
      begin
         MATCH := true;
         for IX := 1 to MX do
         begin
            for JX := 1 to QX do
            begin
               MAT14[IX,JX] := MAT15[IX,JX];
            end;
         end;
      end
      else
         MATCH := false;

end;

PROCEDURE MATPRI(MAT3:vector;ROWS1, COLS2:integer);

{
 Procedure to provide a printout of all calculated matrices if specified by
 descriptor PRIMAT in input file "STRUCT".  Again, subject is only defined
 (MATCH = TRUE) if numbers of rows and columns match
```

Variables used are:-

IY : Parameter for compiling matrix rows to print
JY : Parameter for compiling matrix columns to print
ROWS1 : Number of rows in matrix
COLS2 : Number of columns in matrix
MAT3 : Matrix to be printed
LIMIT1 : Limit on dimensions (rows) of the matrix
LIMIT2 : Limit on dimensions (columns) of the matrix
FORM : Character variable used to print MAT3. It is initialised as '(1X, ££F8.2)' and the repetition indicator ££ is replaced by the value of COLS2 calculated during program execution
MATCH : True if resultant matrix defined (rows and columns match) else false

}

```
var   IY, JY                    :integer;

      MATCH                     :boolean;

begin

   for IY := 1 to ROWS1 do
   begin
      for JY := 1 to COLS2 do
      begin
         writeln(lst,MAT3[IY,JY]);
      end;
   end;

end;

END.
```

```
{$B+}   {Boolean complete evaluation on}
{$S+}   {Stack checking on}
{$I+}   {I/O checking on}
{$N+}   {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT GCMATQ;        {Version 1      Created on 13th June 1991}

{$R+}   {Compiler range checking for arrays, etc.}
{$F+}   {Far call compiler directive}
{$O+}   {Overlay compiler directive}


{
 GCMATQ is an overlaid unit of GCAP03, a Generalised Circuit Analysis Program
 for pulsed power.  This is version 1 of the unit which is used to
 formulate the fundamental cutset matrix Q and fundamental submatrix F
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Printer,
  GcInit,
  GcSort;

var   FA, FB, FC, FD,
      FE, F6, FG, FH,
      FI, FJ, FK, FL,
      FM, FQ, FO, FP,

      HA, HB, HC, HD,
      HE, HF, HG, HH,
      HJ, HK, HL,
      HO, HP                                    :vector;

{procedure matq;
procedure prmatq;
procedure matf;
procedure prmatf;
procedure fsub;
procedure prfsub;    }
procedure matrixq;

Implementation

{-------------------------------------------------------------------------}

{Now declare the procedures and functions}
```

```
PROCEDURE MATRIXQ;

var   XB, XC, XD, WB, WC, WD,
      YE, YI, YM,
      Y1, Y2, Y3                :integer;

      Q                 :array [1..max_num_branch,1..max_num_link] of integer;
      F                 :array [1..max_num_branch,1..max_num_branch] of integer;

{ Number of tree branch independent voltage sources  : W
  Number of tree branch capacitances                 : DD
  Number of tree branch conductances                 : BB
  Number of tree branch reciprocal inductances       : EE
  Number of link reciprocal capacitances (elastances) : U
  Number of link resistances                         : X
  Number of link inductances                         : V
  Number of link independent current sources         : AA   }


PROCEDURE MATQ;

{
 Procedure to construct the fundamental cutset matrix Q from the input file
 "COMPON" and the proper tree
}

var   a3, b3                                    :integer;

      param                                     :boolean;

begin

     for a3 := 1 to T1 do
     begin
        for b3 := 1 to num1 do
        begin

           param := true;

if ((COMP8[a3,2]=COMP8[b3,2]) or (COMP8[a3,2]=COMP8[b3,3])
 or (COMP8[a3,3]=COMP8[b3,2]) or (COMP8[a3,3]=COMP8[b3,3])) then
              begin
                 param := false;
              end;

           if (not param) then
           begin
                 Q[a3,b3] := 1;
           end
           else
           begin
                 Q[a3,b3] := 0
           end;
```

```
        end;

    end;

end;

PROCEDURE PRMATQ;

{
 Procedure to print the fundamental cutset matrix, Q
}

var   a, b                                          :integer;

begin

    for a := 1 to T1 do
    begin
       for b := 1 to num1 do
       begin
          writeln(lst,'Q[',a,',',b,'] = ',Q[a,b]);
       end;
    end;
end;

PROCEDURE MATF;

{
 Procedure to formulate the fundamental F submatrix, derived from the
 fundamental cutset matrix Q
}

var   a3, b3                                        :integer;

begin

    for a3 := 1 to T1 do
    begin
       for b3 := 1 to (num1-T1) do
       begin
          F[a3,b3] := Q[a3,b3+T1];
       end;
    end;

end;

PROCEDURE PRMATF;

{
 Procedure to print the fundamental F submatrix
}

var   a, b                                          :integer;

begin
```

```
      for a := 1 to T1 do
      begin
         for b := 1 to (num1-T1) do
         begin
            writeln(lst,'F[',a,',',b,'] = ',F[a,b]);
         end;
      end;
end;

PROCEDURE FSUB;

{
 Procedure to identify the F-submatrices Fvs, Fvr, etc. (and F'vs, F'vr, etc.)
 by partitioning the fundamental F-matrix
}

var   IA, IB, IC, ID, IE, I6,
      IG, IH, II, IJ, IK, IL,
      IM, IQ, IO, IP,
      JA, JB, JC, JD, JE, JF,
      JG, JH, JI, JJ, JK, JL,
      JM, JN, JO, JP                                    :integer;

begin

{ Identify matrices FA (=Fvs) and HA (=F'vs) }

      if ((U <> 0) and (W <> 0)) then
      begin
         for IA := 1 to W do
         begin
            for JA := 1 to U do
            begin
               FA[IA,JA] := F[IA,JA];
               HA[JA,IA] := F[IA,JA];
            end;
         end;
      end;

{ Identify matrices FB (=Fvr) and HB (=F'vr) }

      XB := U + 1;
      WB := U + X;

      if ((W <> 0) and (X <> 0)) then
      begin
         for IB := 1 to W do
         begin
            for JB := XB to WB do
            begin
               FB[IB,JB-U] := F[IB,JB];
               HB[JB-U,IB] := F[IB,JB-U];
            end;
         end;
```

```
      end;

{ Identify matrices FC (=Fvl) and HC (=F'vl) }

      XC := U + X + 1;
      WC := U + X + V;

      if ((W <> 0) and (V <> 0)) then
      begin
         for IC := 1 to W do
         begin
            for JC := XC to WC do
            begin
               FC[IC,JC-XC+1] := F[IC,JC];
               HC[JC-XC+1,IC] := F[IC,JC-XC+1];
            end;
         end;
      end;

{ Identify matrices FD (=Fvi) and HD (=F'vi) }

      XD := U + X + V + 1;
      WD := U + X + V + AA;

      if ((W <> 0) and (AA <> 0)) then
      begin
         for ID := 1 to W do
         begin
            for JD := XD to WD do
            begin
               FD[ID,JD-XD+1] := F[ID,JD];
               HD[JD-XD+1,ID] := F[ID,JD-XD+1];
            end;
         end;
      end;

{ Identify matrices FE (=Fcs) and HE (=F'cs) }

      YE := W + 1;
      Y1 := W + DD;

      if ((U <> 0) and (DD <> 0)) then
      begin
         for IE := YE to Y1 do
         begin
            for JE := 1 to U do
            begin
               FE[IE-W,JE] := F[IE,JE];
               HE[JE,IE-W] := F[IE-W,JE];
            end;
         end;
      end;

{ Identify matrices F6 (=Fcr) and HF (=F'cr) }
```

```
      if ((X <> 0) and (DD <> 0)) then
      begin
         for I6 := YE to Y1 do
         begin
            for JF := XB to WB do
            begin
               F6[I6-W,JF-U] := F[I6,JF];
               HF[JF-U,I6-W] := F[I6-W,JF-U];
            end;
         end;
      end;

{ Identify matrices FG (=Fcl) and HG (=F'cl) }

      if ((V <> 0) and (DD <> 0)) then
      begin
         for IG := YE to Y1 do
         begin
            for JG := XC to WC do
            begin
               FG[IG-YE+1,JG-XC+1] := F[IG,JG];
               HG[JG-XC+1,IG-YE+1] := F[IG-YE+1,JG-XC+1];
            end;
         end;
      end;

{ Identify matrices FH (=Fci) and HH (=F'ci) }

      if ((AA <> 0) and (DD <> 0)) then
      begin
         for IH := YE to Y1 do
         begin
            for JH := XD to WD do
            begin
               FH[IH-W,JH-XD+1] := F[IH,JH];
               HH[JH-XD+1,IH-W] := F[IH-W,JH-XD+1];
            end;
         end;
      end;

{ Identify matrix FI (=Fgs) }

      YI := W + DD + 1;
      Y2 := W + DD + BB;

      if ((U <> 0) and (BB <> 0)) then
      begin
         for II := YI to Y2 do
         begin
            for JI := 1 to U do
            begin
               FI[II-Y1,JI] := F[II,JI];
            end;
         end;
      end;
```

```
{ Identify matrices FJ (=Fgr) and HJ (=F'gr) }

      if ((X <> 0) and (BB <> 0)) then
      begin
         for IJ := YI to Y2 do
         begin
            for JJ := XB to WB do
            begin
               FJ[IJ-Y1,JJ-U] := F[IJ,JJ];
               HJ[JI,II-Y1] := F[II-Y1,JI];
            end;
         end;
      end;

{ Identify matrices FK (=Fgl) and HK (=F'gl) }

      if ((V <> 0) and (BB <> 0)) then
      begin
         for IK := YI to Y2 do
         begin
            for JK := XC to WC do
            begin
               FK[IK-YI+1,JK-XC+1] := F[IK,JK];
               HK[JK-XC+1,IK-YI+1] := F[IK-YI+1,JK-XC+1];
            end;
         end;
      end;

{ Identify matrices FL (=Fgi) and HL (=F'gi) }

      if ((AA <> 0) and (BB <> 0)) then
      begin
         for IL := YI to Y2 do
         begin
            for JL := XD to WD do
            begin
               FL[IL-Y1,JL-XD+1] := F[IL,JL];
               HL[JL-XD+1,IL-Y1] := F[IL-Y1,JL-XD+1];
            end;
         end;
      end;

{ Identify matrix FM (=Fts) }

      YM := W + DD + BB + 1;
      Y3 := W + DD + BB + EE;

      if ((U <> 0) and (EE <> 0)) then
      begin
         for IM := YM to Y3 do
         begin
            for JM := 1 to U do
            begin
               FM[IM-Y2,JM] := F[IM,JM];
```

```
            end;
         end;
      end;

{ Identify matrix FQ (=Ftr) }

      if ((X <> 0) and (EE <> 0)) then
      begin
         for IQ := YM to Y3 do
         begin
            for JN := XB to WB do
            begin
               FQ[IQ-Y2,JN-U] := F[IQ,JN];
            end;
         end;
      end;

{ Identify matrices FO (=Ftl) and HO (=F'tl) }

      if ((V <> 0) and (EE <> 0)) then
      begin
         for IO := YM to Y3 do
         begin
            for JO := XC to WC do
            begin
               FO[IO-Y2,JO-XC+1] := F[IO,JO];
               HO[JO-XC+1,IO-Y2] := F[IO-Y2,JO-XC+1];
            end;
         end;
      end;

{ Identify matrices FP (=Fti) and HP (=F'ti) }

      if ((AA <> 0) and (EE <> 0)) then
      begin
         for IP := YM to Y3 do
         begin
            for JP := XD to WD do
            begin
               FP[IP-Y2,JP-XD+1] := F[IP,JP];
               HP[JP-XD+1,IP-Y2] := F[IP-Y2,JP-XD+1];
            end;
         end;
      end;

end;

PROCEDURE PRFSUB;

{
 Procedure to print the F submatrices
}

var   IA, IB, IC, ID, IE, I6,
      IG, IH, II, IJ, IK, IL,
```

```
        IM, IQ, IO, IP,
        JA, JB, JC, JD, JE, JF,
        JG, JH, JI, JJ, JK, JL,
        JM, JN, JO, JP                                  :integer;

begin                   {Print the F-submatrices}

        writeln(lst,'F SUBMATRICES');
        writeln(lst,'==============');

        if ((W <> 0) and (U <> 0)) then
        begin
           writeln(lst);
           writeln(lst,'Fvs');
           writeln(lst,'---');
           for IA := 1 to W do
           begin
              for JA := 1 to U do
              begin
                 writeln(lst,FA[IA,JA]);
              end;
           end;
        end;

        if ((W <> 0) and (X <> 0)) then
        begin
           writeln(lst);
           writeln(lst,'Fvr');
           writeln(lst,'---');
           for IB := 1 to W do
           begin
              for JB := 1 to X do
              begin
                 writeln(lst,FB[IB,JB]);
              end;
           end;
        end;

        if ((W <> 0) and (V <> 0)) then
        begin
           writeln(lst);
           writeln(lst,'Fvl');
           writeln(lst,'---');
           for IC := 1 to W do
           begin
              for JC := 1 to V do
              begin
                 writeln(lst,FC[IC,JC]);
              end;
           end;
        end;

        if ((W <> 0) and (AA <> 0)) then
        begin
           writeln(lst);
```

```
    writeln(lst,'Fvi');
    writeln(lst,'---');
    for ID := 1 to W do
    begin
       for JD := 1 to AA do
       begin
          writeln(lst,FD[ID,JD]);
       end;
    end;
end;

if ((DD <> 0) and (U <> 0)) then
begin
    writeln(lst);
    writeln(lst,'Fcs');
    writeln(lst,'---');
    for IE := 1 to DD do
    begin
       for JE := 1 to U do
       begin
          writeln(lst,FE[IE,JE]);
       end;
    end;
end;

if ((DD <> 0) and (X <> 0)) then
begin
    writeln(lst);
    writeln(lst,'Fcr');
    writeln(lst,'---');
    for I6 := 1 to DD do
    begin
       for JF := 1 to X do
       begin
          writeln(lst,F6[I6,JF]);
       end;
    end;
end;

if ((DD <> 0) and (V <> 0)) then
begin
    writeln(lst);
    writeln(lst,'Fcl');
    writeln(lst,'---');
    for IG := 1 to DD do
    begin
       for JG := 1 to V do
       begin
          writeln(lst,FG[IG,JG]);
       end;
    end;
end;

if ((DD <> 0) and (AA <> 0)) then
begin
```

```
    writeln(lst);
    writeln(lst,'Fci');
    writeln(lst,'---');
    for IH := 1 to DD do
    begin
       for JH := 1 to AA do
       begin
          writeln(lst,FH[IH,JH]);
       end;
    end;
end;

if ((BB <> 0) and (U <> 0)) then
begin
    writeln(lst);
    writeln(lst,'Fgs');
    writeln(lst,'---');
    for II := 1 to BB do
    begin
       for JI := 1 to U do
       begin
          writeln(lst,FI[II,JI]);
       end;
    end;
end;

if ((BB <> 0) and (X <> 0)) then
begin
    writeln(lst);
    writeln(lst,'Fgr');
    writeln(lst,'---');
    for IJ := 1 to BB do
    begin
       for JJ := 1 to X do
       begin
          writeln(lst,FJ[IJ,JJ]);
       end;
    end;
end;

if ((BB <> 0) and (V <> 0)) then
begin
    writeln(lst);
    writeln(lst,'Fgl');
    writeln(lst,'---');
    for IK := 1 to BB do
    begin
       for JK := 1 to V do
       begin
          writeln(lst,FK[IK,JK]);
       end;
    end;
end;

if ((BB <> 0) and (AA <> 0)) then
```

```
begin
   writeln(lst);
   writeln(lst,'Fgi');
   writeln(lst,'---');
   for IL := 1 to BB do
   begin
      for JL := 1 to AA do
      begin
         writeln(lst,FL[IL,JL]);
      end;
   end;
end;

if ((EE <> 0) and (U <> 0)) then
begin
   writeln(lst);
   writeln(lst,'Fts');
   writeln(lst,'---');
   for IM := 1 to EE do
   begin
      for JM := 1 to U do
      begin
         writeln(lst,FM[IM,JM]);
      end;
   end;
end;

if ((EE <> 0) and (X <> 0)) then
begin
   writeln(lst);
   writeln(lst,'Ftr');
   writeln(lst,'---');
   for IQ := 1 to EE do
   begin
      for JN := 1 to X do
      begin
         writeln(lst,FQ[IQ,JN]);
      end;
   end;
end;

if ((EE <> 0) and (V <> 0)) then
begin
   writeln(lst);
   writeln(lst,'Ftl');
   writeln(lst,'---');
   for IO := 1 to EE do
   begin
      for JO := 1 to V do
      begin
         writeln(lst,FO[IO,JO]);
      end;
   end;
end;
```

```
if ((EE <> 0) and (AA <> 0)) then
begin
   writeln(lst);
   writeln(lst,'Fti');
   writeln(lst,'---');
   for IP := 1 to EE do
   begin
      for JP := 1 to AA do
      begin
         writeln(lst,FP[IP,JP]);
      end;
   end;
end;

{Print the F transpose sub-matrices}

writeln(lst,'F TRANSPOSE SUB-MATRICES');
writeln(lst,'========================');

if ((W <> 0) and (U <> 0)) then
begin
   writeln(lst);
   writeln(lst,'F/vs');
   writeln(lst,'----');
   for JA := 1 to U do
   begin
      for IA := 1 to W do
      begin
         writeln(lst,HA[JA,IA]);
      end;
   end;
end;

if ((W <> 0) and (X <> 0)) then
begin
   writeln(lst);
   writeln(lst,'F/vr');
   writeln(lst,'----');
   for JB := 1 to X do
   begin
      for IB := 1 to W do
      begin
         writeln(lst,HB[JB,IB]);
      end;
   end;
end;

if ((W <> 0) and (V <> 0)) then
begin
   writeln(lst);
   writeln(lst,'F/vl');
   writeln(lst,'----');
   for JC := 1 to V do
   begin
      for IC := 1 to W do
```

```
        begin
            writeln(lst,HC[JC,IC]);
        end;
    end;
end;

if ((W <> 0) and (AA <> 0)) then
begin
    writeln(lst);
    writeln(lst,'F/vi');
    writeln(lst,'----');
    for JD := 1 to AA do
    begin
        for ID := 1 to W do
        begin
            writeln(lst,HD[JD,ID]);
        end;
    end;
end;

if ((DD <> 0) and (U <> 0)) then
begin
    writeln(lst);
    writeln(lst,'F/cs');
    writeln(lst,'----');
    for JE := 1 to U do
    begin
        for IE := 1 to DD do
        begin
            writeln(lst,HE[JE,IE]);
        end;
    end;
end;

if ((DD <> 0) and (X <> 0)) then
begin
    writeln(lst);
    writeln(lst,'F/cr');
    writeln(lst,'----');
    for JF := 1 to X do
    begin
        for I6 := 1 to DD do
        begin
            writeln(lst,HF[JF,I6]);
        end;
    end;
end;

if ((DD <> 0) and (V <> 0)) then
begin
    writeln(lst);
    writeln(lst,'F/cl');
    writeln(lst,'----');
    for JG := 1 to V do
    begin
```

```
        for IG := 1 to DD do
        begin
            writeln(lst,HG[JG,IG]);
        end;
    end;
end;

if ((DD <> 0) and (AA <> 0)) then
begin
    writeln(lst);
    writeln(lst,'F/ci');
    writeln(lst,'----');
    for JH := 1 to AA do
    begin
        for IH := 1 to DD do
        begin
            writeln(lst,HH[JH,IH]);
        end;
    end;
end;

if ((BB <> 0) and (X <> 0)) then
begin
    writeln(lst);
    writeln(lst,'F/gr');
    writeln(lst,'----');
    for JJ := 1 to X do
    begin
        for IJ := 1 to BB do
        begin
            writeln(lst,HJ[JJ,IJ]);
        end;
    end;
end;

if ((BB <> 0) and (V <> 0)) then
begin
    writeln(lst);
    writeln(lst,'F/gl');
    writeln(lst,'----');
    for JK := 1 to V do
    begin
        for IK := 1 to BB do
        begin
            writeln(lst,HK[JK,IK]);
        end;
    end;
end;

if ((BB <> 0) and (AA <> 0)) then
begin
    writeln(lst);
    writeln(lst,'F/gi');
    writeln(lst,'----');
    for JL := 1 to AA do
```

```
        begin
          for IL := 1 to BB do
          begin
            writeln(lst,HL[JL,IL]);
          end;
        end;
      end;

      if ((EE <> 0) and (V <> 0)) then
      begin
        writeln(lst);
        writeln(lst,'F/tl');
        writeln(lst,'---');
        for JO := 1 to V do
        begin
          for IO := 1 to EE do
          begin
            writeln(lst,HO[JO,IO]);
          end;
        end;
      end;

      if ((EE <> 0) and (AA <> 0)) then
      begin
        writeln(lst);
        writeln(lst,'F/ti');
        writeln(lst,'---');
        for JP := 1 to AA do
        begin
          for IP := 1 to EE do
          begin
            writeln(lst,HP[JP,IP]);
          end;
        end;
      end;


end;


begin                 {Start of code section for procedure MATRIXQ}

    matq;
    prmatq;           {Print the fundamental cutset matrix Q}
    matf;
    prmatf;           {Print the fundamental F submatrix}
    fsub;
    prfsub;           {Print the F submatrices}

end;

END.
```

```
{$B+}   {Boolean complete evaluation on}
{$S+}   {Stack checking on}
{$I+}   {I/O checking on}
{$N+}   {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT GCSTAT;        {Version 1      Created on 11th June 1991}

{$R+}   {Compiler range checking for arrays, etc.}
{$F+}   {Far call compiler directive}
{$O+}   {Overlay compiler directive}


{
 GCSTAT is an overlaid unit of GCAP03, a Generalised Circuit Analysis Program
 for pulsed power.  This is version 1 of the unit which is used to
 formulate the state matrices A, B, C, D and E
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Printer,
  GcInit,
  GcSort,
  GcMat,
  GcMatq;

var     A               :array[1..max_num_AX,1..max_num_AX] of real;
        B               :array[1..max_num_AX,1..max_num_BX] of real;
        AI              :array[1..max_num_AX,1..max_num_AX] of real;
        C               :array[1..max_num_CX,1..max_num_CX] of real;
        D               :array[1..max_num_DX,1..max_num_DX] of real;
        E               :array[1..max_num_EX,1..max_num_EX] of real;


{procedure elem;
procedure inv1;
procedure int1;
procedure inv2;
procedure hybr;
procedure stat; }
procedure statem;

Implementation

{-------------------------------------------------------------------------}

{Now declare the procedures and functions}
```

```
PROCEDURE STATEM;

var   CC, GG, LT,
      CS, RR, LL,

      RI, GI,

      UA, UB, UC, UD,
      UE, UF, UG, UH,

      CA, L, R, G,

      CJ, RJ, GJ, LJ,

      H1, H2, H3, H4,
      H5, H6, H7, H8, H0,

      W2, W3, W5, W7,
      W8, W9, W0,

      YU, YV, YW,

      Y11, Y12, Y13, Y14,

      H9, Y5, W1, Y6,
      Y7, W4, Y8, W6,

      TA, TB, TC, TD,
      TE, TF, TG, TH,
      TI, TJ, TK, TL,
      TM, TN, TP, TQ,
      TX, TZ, TY, TU,

      RA, RB, RC, RD,
      RE, RF, RG, RH,
      RJ, RK, RL, RM,
      RN, RO, RP, RQ,
      RS, RT, RU, RV,
      RW, RX, RY, RZ,
      R1, R2, R3, R4,
      R5, R6, R7, R8,

      SA, SB, SC, SD,
      SE, SF, SG, SH,
      SJ, SK, SL, SM,
      SN, SO, SP, SQ,
      SS, ST, SU, SV,
      SW, SX, SY, SZ,
      S1, S2, S3, S4,
      S5, S6, S7, S8,
      S9, S10, S11, S12,
      S13, S14, S15, S16,
      S17, S18, S19, S20,
```

```
      P1, P2, P3, P4,
      P5, P6, P7, P8,
      P9, P10, P11, P12                                  :vector;

PROCEDURE ELEM;

{
 Procedure to calculate the Element matrices Cc, Cs, Gg, Rr, Lt and Ll
 }

var   K1, K2, K3, K4, K5, K6,
      x1, x2                                             :integer;

begin

        writeln(lst,'ELEMENT MATRICES');
        writeln(lst,'================');

{
 Element matrix Cc (tree branch capacitances)
 }

     if (DD <> 0) then
     begin
        writeln(lst,'Cc');
        writeln(lst,'--');
        matzer(CC,DD,DD,DD,DD);
        for x1 := 1 to DD do
        begin
           for x2 := 1 to DD do
           begin
              writeln(lst,'CC[',x1,',',x2,'] = ',ZERO[x1,x2]:11);
           end;
        end;
        for K1 := 1 to DD do
        begin
           CC[K1,K1] := COMP8[W+K1,1];
           writeln(lst,'Cc[',K1,',',K1,'] = ',CC[K1,K1]);
        end;
     end;

{
 Element matrix Cs (link elastances)
 }

     if (U <> 0) then
     begin
        writeln(lst,'Cs');
        writeln(lst,'--');
        matzer(CS,U,U,U,U);
        for x1 := 1 to U do
        begin
           for x2 := 1 to U do
```

```
            begin
               writeln(lst,'CS[',x1,',',x2,'] = ',ZERO[x1,x2]:11);
            end;
         end;
         for K2 := 1 to U do
         begin
            CS[K2,K2] := COMP8[W+DD+K2,1];
            writeln(lst,'Cs[',K2,',',K2,'] = ',CS[K2,K2]);
         end;
      end;

{
 Element matrix Gg (tree branch conductances)
 }

      if (BB <> 0) then
      begin
         writeln(lst,'Gg');
         writeln(lst,'--');
         matzer(GG,BB,BB,BB,BB);
         for x1 := 1 to BB do
         begin
            for x2 := 1 to BB do
            begin
               writeln(lst,'GG[',x1,',',x2,'] = ',ZERO[x1,x2]:11);
            end;
         end;
         for K3 := 1 to BB do
         begin
            GG[K3,K3] := COMP8[W+DD+U+K3,1];
            writeln(lst,'Gg[',K3,',',K3,'] = ',GG[K3,K3]);
         end;
      end;

{
 Element matrix Rr (link resistances)
 }

      if (X <> 0) then
      begin
         writeln(lst,'Rr');
         writeln(lst,'--');
         matzer(RR,X,X,X,X);
         for x1 := 1 to X do
         begin
            for x2 := 1 to X do
            begin
               writeln(lst,'RR[',x1,',',x2,'] = ',ZERO[x1,x2]:11);
            end;
         end;
         for K4 := 1 to X do
         begin
            RR[K4,K4] := COMP8[W+DD+U+BB+K4,1];
            writeln(lst,'Rr[',K4,',',K4,'] = ',RR[K4,K4]);
         end;
```

```
    end;

{
 Element matrix Lt (tree branch reciprocal inductances)
 }

    if (EE <> 0) then
    begin
       writeln(lst,'Lt');
       writeln(lst,'--');
       matzer(LT,EE,EE,EE,EE);
       for x1 := 1 to EE do
       begin
          for x2 := 1 to EE do
          begin
             writeln(lst,'LT[',x1,',',x2,'] = ',ZERO[x1,x2]:11);
          end;
       end;
       for K5 := 1 to EE do
       begin
          LT[K5,K5] := COMP8[W+DD+U+BB+X+K5,1];
          writeln(lst,'Lt[',K5,',',K5,'] = ',LT[K5,K5]);
       end;
    end;

{
 Element matrix Ll (link inductances)
 }

    if (V <> 0) then
    begin
       writeln(lst,'Ll');
       writeln(lst,'--');
       matzer(LL,V,V,V,V);
       for x1 := 1 to V do
       begin
          for x2 := 1 to V do
          begin
             writeln(lst,'LL[',x1,',',x2,'] = ',ZERO[x1,x2]:11);
          end;
       end;
       for K6 := 1 to V do
       begin
          LL[K6,K6] := COMP8[W+DD+U+BB+X+EE+K6,1];
          writeln(lst,'Ll[',K6,',',K6,'] = ',LL[K6,K6]);
       end;
    end;

end;

PROCEDURE INV1;

{
 Procedure to calculate the Inverse matrices of Rr (Rr-1) and Gg (Gg-1)
 }
```

```
var   x1, x2                                :integer;

begin

{
 Inverse matrix Gg-1 (tree branch conductances)
 }

     if (BB <> 0) then
     begin
        matinv(GG,BB,BB,BB,BB);
        for x1 := 1 to BB do
        begin
           for x2 := 1 to BB do
           begin
              writeln(lst,'GG[',x1,',',x2,'] = ',GG[x1,x2]:11);
              GI[x1,x2] := INVS[x1,x2];
              writeln(lst,'GI[',x1,',',x2,'] = ',GI[x1,x2]:11);
           end;
        end;
     end;

{
 Inverse matrix Rr-1 (link resistances)
 }

     if (X <> 0) then
     begin
        matinv(RR,X,X,X,X);
        for x1 := 1 to X do
        begin
           for x2 := 1 to X do
           begin
              writeln(lst,'RR[',x1,',',x2,'] = ',RR[x1,x2]:11);
              RI[x1,x2] := INVS[x1,x2];
              writeln(lst,'RI[',x1,',',x2,'] = ',RI[x1,x2]:11);
           end;
        end;
     end;
end;


PROCEDURE INT1;

{
 Procedure to calculate the first set of Intermediate matrices UA -> UH,
 aswell as the matrices CA, L, R and G
 }

var   x1, x2                                :integer;

begin

        writeln(lst,'INTERMEDIATE MATRICES');
```

```
        writeln(lst,'====================');

{ Matrix CA (via intermediate matrices UA and UB and element matrix Cc) }

    if ((DD <> 0) and (U <> 0)) then
    begin
      matmul(CS,HE,U,U,U,DD);
      for x1 := 1 to U do
      begin
        for x2 := 1 to DD do
        begin
          writeln(lst,'CS[',x1,',',x2,'] = ',CS[x1,x2]:11);
          writeln(lst,'HE[',x1,',',x2,'] = ',HE[x1,x2]:11);
          UA[x1,x2] := PROD[x1,x2];
          writeln(lst,'UA[',x1,',',x2,'] = ',UA[x1,x2]:11);
        end;
      end;

      matmul(FE,UA,DD,U,U,DD);
      for x1 := 1 to DD do
      begin
        for x2 := 1 to DD do
        begin
          writeln(lst,'FE[',x1,',',x2,'] = ',FE[x1,x2]:11);
          writeln(lst,'UA[',x1,',',x2,'] = ',UA[x1,x2]:11);
          UB[x1,x2] := PROD[x1,x2];
          writeln(lst,'UB[',x1,',',x2,'] = ',UB[x1,x2]:11);
        end;
      end;

      matplu(CC,UB,DD,DD,DD,DD);
      for x1 := 1 to DD do
      begin
        for x2 := 1 to DD do
        begin
          writeln(lst,'CC[',x1,',',x2,'] = ',CC[x1,x2]:11);
          writeln(lst,'UB[',x1,',',x2,'] = ',UB[x1,x2]:11);
          CA[x1,x2] := PLUS[x1,x2];
          writeln(lst,'CA[',x1,',',x2,'] = ',CA[x1,x2]:11);
        end;
      end;

      writeln(lst,'CAP');
      writeln(lst,'---');
      matpri(CA,DD,DD);
      writeln(lst);
    end;

{ Matrix L (via intermediate matrices UC and UD and element matrix Ll) }

    if ((V <> 0) and (EE <> 0)) then
    begin
      matmul(LT,FO,EE,EE,EE,V);
      for x1 := 1 to EE do
      begin
```

```
            for x2 := 1 to V do
            begin
               UC[x1,x2] := PROD[x1,x2];
            end;
         end;

         matmul(HO,UC,V,EE,EE,V);
         for x1 := 1 to V do
         begin
            for x2 := 1 to V do
            begin
               UD[x1,x2] := PROD[x1,x2];
            end;
         end;

         matplu(LL,UD,V,V,V,V);
         for x1 := 1 to V do
         begin
            for x2 := 1 to V do
            begin
               L[x1,x2] := PLUS[x1,x2];
            end;
         end;

         writeln(lst,'IND');
         writeln(lst,'---');
         matpri(L,V,V);
         writeln(lst);
      end;

{ Matrix R (via intermediate matrices UE and UF and element matrix Rr) }

      if ((X <> 0) and (BB <> 0)) then
      begin
         matmul(GI,FJ,BB,BB,BB,X);
         for x1 := 1 to BB do
         begin
            for x2 := 1 to X do
            begin
               UE[x1,x2] := PROD[x1,x2];
            end;
         end;

         matmul(HJ,UE,X,BB,BB,X);
         for x1 := 1 to X do
         begin
            for x2 := 1 to X do
            begin
               UF[x1,x2] := PROD[x1,x2];
            end;
         end;

         matplu(RR,UF,X,X,X,X);
         for x1 := 1 to X do
         begin
```

```
            for x2 := 1 to X do
            begin
               R[x1,x2] := PLUS[x1,x2];
            end;
         end;

         writeln(lst,'RES');
         writeln(lst,'---');
         matpri(R,X,X);
         writeln(lst);
      end;

{ Matrix G (via intermediate matrices UG and UH and element matrix Gg) }

      if ((X <> 0) and (BB <> 0)) then
      begin
         matmul(RI,HJ,X,X,X,BB);
         for x1 := 1 to X do
         begin
            for x2 := 1 to BB do
            begin
               UG[x1,x2] := PROD[x1,x2];
            end;
         end;

         matmul(FJ,UG,BB,X,X,BB);
         for x1 := 1 to BB do
         begin
            for x2 := 1 to BB do
            begin
               UH[x1,x2] := PROD[x1,x2];
            end;
         end;

         matplu(GG,UH,BB,BB,BB,BB);
         for x1 := 1 to BB do
         begin
            for x2 := 1 to BB do
            begin
               G[x1,x2] := PLUS[x1,x2];
            end;
         end;

         writeln(lst,'CON');
         writeln(lst,'---');
         matpri(G,BB,BB);
         writeln(lst);
      end;

end;

PROCEDURE INV2;

{ Procedure to calculate the Inverse matrices of CA (C-1), R (R-1), G (G-1)
 and L (L-1) }
```

```
var   x1, x2                                  :integer;

begin

        writeln(lst,'INVERSE MATRICES');
        writeln(lst,'================');

{ Inverse matrix R-1 }

     if (X <> 0) then
     begin
        matinv(R,X,X,X,X);
        for x1 := 1 to X do
        begin
           for x2 := 1 to X do
           begin
              RJ[x1,x2] := INVS[x1,x2];
           end;
        end;

        writeln(lst,'R-1');
        writeln(lst,'---');
        matpri(RJ,X,X);
        writeln(lst);
     end;

{ Inverse matrix G-1 }

     if (BB <> 0) then
     begin
        matinv(G,BB,BB,BB,BB);
        for x1 := 1 to BB do
        begin
           for x2 := 1 to BB do
           begin
              GJ[x1,x2] := INVS[x1,x2];
           end;
        end;

        writeln(lst,'G-1');
        writeln(lst,'---');
        matpri(GJ,BB,BB);
        writeln(lst);
     end;

{ Inverse matrix C-1 }

     if (DD <> 0) then
     begin
        matinv(CA,DD,DD,DD,DD);
        for x1 := 1 to DD do
        begin
           for x2 := 1 to DD do
           begin
```

```
            CJ[x1,x2] := INVS[x1,x2];
         end;
      end;

      writeln(lst,'C-1');
      writeln(lst,'---');
      matpri(CJ,DD,DD);
      writeln(lst);
   end;

{ Inverse matrix L-1 }

   if (V <> 0) then
   begin
      matinv(L,V,V,V,V);
      for x1 := 1 to V do
      begin
         for x2 := 1 to V do
         begin
            LJ[x1,x2] := INVS[x1,x2];
         end;
      end;

      writeln(lst,'L-1');
      writeln(lst,'---');
      matpri(LJ,V,V);
      writeln(lst);
   end;

end;

PROCEDURE HYBR;

{ Procedure to calculate the Hybrid matrices Hcc, Hcl, Hcv, Hci, Hlc, Hll,
 Hlv and Hli }

var   x1, x2                              :integer;

begin

{ Hybrid matrix Hcc (= H9 (via H1)) }

   if ((DD <> 0) and (X <> 0)) then
   begin
      matmul(RJ,HF,X,X,X,DD);
      for x1 := 1 to X do
      begin
         for x2 := 1 to DD do
         begin
            H1[x1,x2] := PROD[x1,x2];
         end;
      end;

      matmul(F6,H1,DD,X,X,DD);
      for x1 := 1 to DD do
```

```
        begin
          for x2 := 1 to DD do
          begin
            H9[x1,x2] := PROD[x1,x2];
          end;
        end;

        matmul(RI,HF,X,X,X,DD);
        for x1 := 1 to X do
        begin
          for x2 := 1 to DD do
          begin
            H5[x1,x2] := PROD[x1,x2];
          end;
        end;

      end;

{ Hybrid matrix Hcl                    }

      if ((DD <> 0) and (BB <> 0) and (X <> 0)) then
      begin
        matmul(FJ,H5,BB,X,X,DD);
        for x1 := 1 to BB do
        begin
          for x2 := 1 to DD do
          begin
            W3[x1,x2] := PROD[x1,x2];
          end;
        end;

        matmul(GJ,W3,BB,BB,BB,DD);
        for x1 := 1 to BB do
        begin
          for x2 := 1 to DD do
          begin
            W9[x1,x2] := PROD[x1,x2];
          end;
        end;

      end;

      if (V <> 0) then
      begin
        matmul(HK,W9,V,BB,BB,DD);
        for x1 := 1 to V do
        begin
          for x2 := 1 to DD do
          begin
            YW[x1,x2] := PROD[x1,x2];
          end;
        end;

        matmin(YW,HG,V,DD,V,DD);
        for x1 := 1 to V do
```

```
    begin
      for x2 := 1 to DD do
      begin
        Y7[x1,x2] := MINU[x1,x2];
      end;
    end;

end;

if ((V <> 0) and (BB <> 0)) then
begin
    matmul(GJ,FK,BB,BB,BB,V);
    for x1 := 1 to BB do
    begin
      for x2 := 1 to V do
      begin
        H6[x1,x2] := PROD[x1,x2];
      end;
    end;

    matmul(HK,H6,V,BB,BB,V);
    for x1 := 1 to V do
    begin
      for x2 := 1 to V do
      begin
        W4[x1,x2] := PROD[x1,x2];
      end;
    end;

    matmul(GI,FK,BB,BB,BB,V);
    for x1 := 1 to BB do
    begin
      for x2 := 1 to V do
      begin
        H2[x1,x2] := PROD[x1,x2];
      end;
    end;

end;

if (X <> 0) then
begin
    matmul(HJ,H2,X,BB,BB,V);
    for x1 := 1 to X do
    begin
      for x2 := 1 to V do
      begin
        HO[x1,x2] := PROD[x1,x2];
      end;
    end;

    matmul(RJ,HO,X,X,X,V);
    for x1 := 1 to X do
    begin
      for x2 := 1 to V do
```

```
            begin
               W7[x1,x2] := PROD[x1,x2];
            end;
         end;

      end;

      if (DD <> 0) then
      begin
         matmul(F6,W7,DD,X,X,V);
         for x1 := 1 to DD do
         begin
            for x2 := 1 to V do
            begin
               YU[x1,x2] := PROD[x1,x2];
            end;
         end;

{          matmin(FG,YU,DD,V,DD,V);}
      end;

      if ((W <> 0) and (X <> 0)) then
      begin
         matmul(RJ,HB,X,X,X,W);
         for x1 := 1 to X do
         begin
            for x2 := 1 to W do
            begin
               H3[x1,x2] := PROD[x1,x2];
            end;
         end;

      end;

      if (DD <> 0) then
      begin
         matmul(F6,H3,DD,X,X,W);
         for x1 := 1 to DD do
         begin
            for x2 := 1 to W do
            begin
               W1[x1,x2] := PROD[x1,x2];
            end;
         end;

         matmul(RI,H8,X,X,X,W);
         for x1 := 1 to X do
         begin
            for x2 := 1 to W do
            begin
               H7[x1,x2] := PROD[x1,x2];
            end;
         end;

      end;
```

```
if (BB <> 0) then
begin
   matmul(FJ,H7,BB,X,X,W);
   for x1 := 1 to BB do
   begin
      for x2 := 1 to W do
      begin
         W5[x1,x2] := PROD[x1,x2];
      end;
   end;

   matmul(GJ,W5,BB,BB,BB,W);
   for x1 := 1 to BB do
   begin
      for x2 := 1 to W do
      begin
         W0[x1,x2] := PROD[x1,x2];
      end;
   end;

end;

if (V <> 0) then
begin
   matmul(HK,W0,V,BB,BB,W);      {Y11 was Y4}
   for x1 := 1 to V do
   begin
      for x2 := 1 to W do
      begin
         Y11[x1,x2] := PROD[x1,x2];
      end;
   end;

end;

if (V <> 0) then
begin
   matmin(Y11,HC,V,V,V,W);
   for x1 := 1 to V do
   begin
      for x2 := 1 to W do
      begin
         Y8[x1,x2] := MINU[x1,x2];
      end;
   end;

end;

if ((AA <> 0) and (BB <> 0)) then
begin
   matmul(GJ,FL,BB,BB,BB,AA);
   for x1 := 1 to BB do
   begin
      for x2 := 1 to AA do
```

```
      begin
         H8[x1,x2] := PROD[x1,x2];
      end;
   end;

end;

if (V <> 0) then
begin
   matmul(HK,H8,V,BB,BB,AA);
   for x1 := 1 to V do
   begin
      for x2 := 1 to AA do
      begin
         W6[x1,x2] := PROD[x1,x2];
      end;
   end;

   matmul(GI,FL,BB,BB,BB,AA);
   for x1 := 1 to BB do
   begin
      for x2 := 1 to AA do
      begin
         H4[x1,x2] := PROD[x1,x2];
      end;
   end;

end;

if (X <> 0) then
begin
   matmul(HJ,H4,X,BB,BB,AA);
   for x1 := 1 to X do
   begin
      for x2 := 1 to AA do
      begin
         W2[x1,x2] := PROD[x1,x2];
      end;
   end;

   matmul(RJ,W2,X,X,X,AA);
   for x1 := 1 to X do
   begin
      for x2 := 1 to AA do
      begin
         W8[x1,x2] := PROD[x1,x2];
      end;
   end;

end;

if (DD <> 0) then
begin
   matmul(F6,W8,DD,X,X,AA);
   for x1 := 1 to DD do
```

```
        begin
          for x2 := 1 to AA do
          begin
            YV[x1,x2] := PROD[x1,x2];
          end;
        end;

      end;

      if (DD <> 0) then
      begin
        matmin(FH,YV,DD,AA,DD,AA);
        for x1 := 1 to DD do
        begin
          for x2 := 1 to AA do
          begin
            Y6[x1,x2] := PROD[x1,x2];
          end;
        end;

      end;

{ Hybrid matrix Hlc (= Y7 (via H5, W3, W9 and YW)) }

{      if ((DD <> 0) and (X <> 0)) then
      begin
        matmul(       );
        if (BB <> 0) then
        begin
          matmul(       );

      end;
 }

       writeln(lst,'HYBRID MATRICES');
       writeln(lst,'================');

       if ((W <> 0) and (U <> 0)) then
       begin
         writeln(lst);
         writeln(lst,'Hcc');
         writeln(lst,'---');
         matpri(H9,DD,DD);
         writeln(lst);
       end;

       if ((W <> 0) and (U <> 0)) then
       begin
         writeln(lst);
         writeln(lst,'Hcl');
         writeln(lst,'---');
         matpri(Y5,DD,V);
         writeln(lst);
       end;
```

```
        if ((W <> 0) and (U <> 0)) then
        begin
            writeln(lst);
            writeln(lst,'Hcv');
            writeln(lst,'---');
            matpri(W1,DD,W);
            writeln(lst);
        end;

        if ((W <> 0) and (U <> 0)) then
        begin
            writeln(lst);
            writeln(lst,'Hci');
            writeln(lst,'---');
            matpri(Y6,DD,AA);
            writeln(lst);
        end;

        if ((W <> 0) and (U <> 0)) then
        begin
            writeln(lst);
            writeln(lst,'Hlc');
            writeln(lst,'---');
            matpri(Y7,V,DD);
            writeln(lst);
        end;

        if ((W <> 0) and (U <> 0)) then
        begin
            writeln(lst);
            writeln(lst,'Hll');
            writeln(lst,'---');
            matpri(W4,V,V);
            writeln(lst);
        end;

        if ((W <> 0) and (U <> 0)) then
        begin
            writeln(lst);
            writeln(lst,'Hlv');
            writeln(lst,'---');
            matpri(Y8,V,W);
            writeln(lst);
        end;

        if ((W <> 0) and (U <> 0)) then
        begin
            writeln(lst);
            writeln(lst,'Hli');
            writeln(lst,'---');
            matpri(W6,V,AA);
            writeln(lst);
        end;

end;
```

```
PROCEDURE STAT;

{ Procedure to calculate the State matrices A, B, C, D and E }

var   x1, x2,
      A1, A2, A3,
      A4, A5, A6,
      A7, A8, A9,
      AA, AB, AC,
      AD, AE,
      B1, B2, B3,
      B4, B5, B6,
      B7, B8, B9,
      BA, BB, BC,
      BD, BF,
      C1, C2, C3,
      C4, C5, C6,
      C7, C8, C9,
      CA, CB, CC,
      CD, CE,
      D1, D2, D3,
      D4, D5, D6,
      D7, D8, D9,
      DA, DB, DC,
      DD, DE,
      E1, E2, E3,
      E4, E5, E6,
      E7, E8, E9,
      EA, EB, EC,
      ED, EE                                    :integer;

begin

     writeln(lst,'STATE MATRICES');
     writeln(lst,'==============');

{ State matrix A }

     if (DD <> 0) then
     begin
        matmul(H9,CJ,DD,DD,DD,DD);
        matmul(HK,W0,V,BB,BB,W);     {Y11 was Y4}
        for x1 := 1 to DD do
        begin
           for x2 := 1 to DD do
           begin
              TA[x1,x2] := PROD[x1,x2];
           end;
        end;
     end;

     if ((DD <> 0) and (V <> 0)) then
     begin
        matmul(Y5,LJ,DD,V,V,V);
```

```
      for x1 := 1 to DD do
      begin
         for x2 := 1 to V do
         begin
            TB[x1,x2] := PROD[x1,x2];
         end;
      end;
   end;

   if ((DD <> 0) and (V <> 0)) then
   begin
      matmul(Y7,CJ,V,DD,DD,DD);
      for x1 := 1 to DD do
      begin
         for x2 := 1 to V do
         begin
            TC[x1,x2] := PROD[x1,x2];
         end;
      end;
   end;

   if (V <> 0) then
   begin
      matmul(W4,LJ,V,V,V,V);
      for x1 := 1 to V do
      begin
         for x2 := 1 to V do
         begin
            TD[x1,x2] := PROD[x1,x2];
         end;
      end;
   end;

{Formulation of A matrix)

   AX := DD + V;

   if (DD <> 0) then
   begin
      for A1 := 1 to DD do
      begin
         for A2 := 1 to DD do
         begin
            AI[A1,A2] := TA[A1,A2];
         end;
      end;
   end;

   A3 := DD+1;

   if (DD <> 0) then
   begin
      for A4 := 1 to DD do
      begin
         for A5 := A3 to AX do
```

```
      begin
         AB := A5 - DD;
         AI[A4,A5] := TB[A4,AB];
      end;
   end;
end;

if (DD <> 0) then
begin
   for A6 := A3 to AX do
   begin
      for A7 := 1 to DD do
      begin
         AC := A6 - DD;
         AI[A6,A7] := TC[AC,A7];
      end;
   end;
end;

if (DD <> 0) then
begin
   for A8 := A3 to AX do
   begin
      for A9 := A3 to AX do
      begin
         AD := A8 - DD;
         AE := A9 - DD;
         AI[A8,A9] := TD[AD,AE];
      end;
   end;
end;

matprd(AI,-1);
for x1 := 1 to AX do
begin
   for x2 := 1 to AX do
      begin
         A[x1,x2] := MAT13[x1,x2];
      end;
   end;
end;

writeln(lst,'TD');
writeln(lst,'--');
matpri(TD,V,V);
writeln(lst);

writeln(lst,'TC');
writeln(lst,'--');
matpri(TC,DD,V);
writeln(lst);

writeln(lst,'TB');
writeln(lst,'--');
matpri(TB,DD,V);
```

```
        writeln(lst);

        writeln(lst,'TA');
        writeln(lst,'--');
        matpri(TA,DD,DD);
        writeln(lst);

        writeln(lst,'A');
        writeln(lst,'-');
        matpri(A,AX,AX);
        writeln(lst);

{ State matrix B }

        if ((U <> 0) and (W <> 0)) then
        begin
           matmul(CS,HA,U,U,U,W);
           for x1 := 1 to U do
           begin
              for x2 := 1 to W do
              begin
                 TE[x1,x2] := PROD[x1,x2];
              end;
           end;
        end;

        if ((DD <> 0) and (W <> 0)) then
        begin
           matmul(FE,TE,DD,U,U,W);
           for x1 := 1 to DD do
           begin
              for x2 := 1 to W do
              begin
                 TF[x1,x2] := PROD[x1,x2];
              end;
            end;
        end;

        if ((DD <> 0) and (W <> 0)) then
        begin
           matmul(TA,TF,DD,DD,DD,W);
           for x1 := 1 to DD do
           begin
              for x2 := 1 to W do
              begin
                 TG[x1,x2] := PROD[x1,x2];
              end;
           end;
        end;

        if ((DD <> 0) and (W <> 0)) then
        begin
           matmin(TG,W1,DD,W,DD,W);
           for x1 := 1 to DD do
           begin
```

```
        for x2 := 1 to W do
        begin
            TH[x1,x2] := MINU[x1,x2];
        end;
    end;
end;

if ((EE <> 0) and (A <> 0)) then
begin
    matmul(LT,FP,EE,EE,EE,AA);
    for x1 := 1 to EE do
    begin
        for x2 := 1 to AA do
        begin
            TI[x1,x2] := PROD[x1,x2];
        end;
    end;
end;

if ((V <> 0) and (AA <> 0)) then
begin
    matmul(HO,TI,V,EE,EE,AA);
    for x1 := 1 to V do
    begin
        for x2 := 1 to AA do
        begin
            TJ[x1,x2] := PROD[x1,x2];
        end;
    end;
end;

if ((DD <> 0) and (AA <> 0)) then
begin
    matmul(TB,TJ,DD,V,V,AA);
    for x1 := 1 to V do
    begin
        for x2 := 1 to AA do
        begin
            TK[x1,x2] := PROD[x1,x2];
        end;
    end;
end;

if ((V <> 0) and (W <> 0)) then
begin
    matmul(TC,TF,V,DD,DD,W);
    for x1 := 1 to V do
    begin
        for x2 := 1 to W do
        begin
            TM[x1,x2] := PROD[x1,x2];
        end;
    end;
end;
```

```
if ((V <> 0) and (W <> 0)) then
begin
   matmin(TM,Y8,V,W,V,W);
   for x1 := 1 to V do
   begin
      for x2 := 1 to W do
      begin
         TN[x1,x2] := MINU[x1,x2];
      end;
   end;
end;

if ((V <> 0) and (AA <> 0)) then
begin
   matmul(TD,TJ,V,V,V,AA);
   for x1 := 1 to V do
   begin
      for x2 := 1 to AA do
      begin
         TP[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

{Formulation of B matrix)

   if ((BX <> 0) and (W <> 0)) then
   begin
      for B1 := 1 to DD do
      begin
         for B2 := 1 to W do
         begin
            B[B1,B2] := TH[B1,B2];
         end;
      end;
   end;

   B3 := W + 1;

   if (AA <> 0) then
   begin
      for B4 := 1 to DD do
      begin
         for B5 := B3 to BX do
         begin
            BA := B5 - DD;
            B[B4,B5] := TK[B4,BA];
         end;
      end;
   end;

   if ((V <> 0) and (W <> 0)) then
   begin
      for B6 := A3 to AX do
      begin
```

```
            for B7 := 1 to W do
            begin
               BC := B6 - DD;
               B[B6,B7] := TN[BC,B7];
            end;
         end;
      end;

      if (AA <> 0) then
      begin
         for B8 := A3 to AX do
         begin
            for B9 := B3 to BX do
            begin
               BD := B8 - DD;
               BF := B9 - DD;
               B[B8,B9] := TP[BD,BF];
            end;
         end;
      end;

      writeln(lst,'TP');
      writeln(lst,'--');
      matpri(TP,V,AA);
      writeln(lst);

      writeln(lst,'TN');
      writeln(lst,'--');
      matpri(TN,V,W);
      writeln(lst);

      writeln(lst,'TK');
      writeln(lst,'--');
      matpri(TK,DD,AA);
      writeln(lst);

      writeln(lst,'TH');
      writeln(lst,'--');
      matpri(TH,DD,W);
      writeln(lst);

      writeln(lst,'B');
      writeln(lst,'-');
      matpri(B,AX,BX);
      writeln(lst);

{ State matrix C }

      BX := W + AA;

      if ((U <> 0) and (W <> 0)) then
      begin
         matmul(W9,CJ,U,U,U,W);
         for x1 := 1 to U do
         begin
```

```
      for x2 := 1 to W do
      begin
         RA[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
   matmul(HI,RE,DD,U,U,W);
   for x1 := 1 to DD do
   begin
      for x2 := 1 to W do
      begin
         RB[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
   matmin(HE,RF,DD,DD,DD,W);
   for x1 := 1 to DD do
   begin
      for x2 := 1 to W do
      begin
         RC[x1,x2] := MINU[x1,x2];
      end;
   end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
   matprd(TC,-1,DD,W,DD,W);
   for x1 := 1 to DD do
   begin
      for x2 := 1 to W do
      begin
         RD[x1,x2] := CONS[x1,x2];
      end;
   end;
end;

if ((EE <> 0) and (A <> 0)) then
begin
   matmul(LJ,RD,EE,EE,EE,AA);
   for x1 := 1 to EE do
   begin
      for x2 := 1 to AA do
      begin
         RE[x1,x2] := PROD[x1,x2];
      end;
   end;
end;
```

```
if ((V <> 0) and (AA <> 0)) then
begin
   matmul(UC,RE,V,EE,EE,AA);
   for x1 := 1 to V do
   begin
      for x2 := 1 to AA do
      begin
         RF[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((DD <> 0) and (AA <> 0)) then
begin
   matmul(HM,RF,DD,V,V,AA);
   for x1 := 1 to V do
   begin
      for x2 := 1 to AA do
      begin
         RG[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((V <> 0) and (W <> 0)) then
begin
   matplu(RC,RG,V,DD,DD,W);
   for x1 := 1 to V do
   begin
      for x2 := 1 to W do
      begin
         RH[x1,x2] := PLUS[x1,x2];
      end;
   end;
end;

if ((U <> 0) and (W <> 0)) then
begin
   matmul(H6,LJ,U,U,U,W);
   for x1 := 1 to U do
   begin
      for x2 := 1 to W do
      begin
         RJ[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
   matmul(FI,RJ,DD,U,U,W);
   for x1 := 1 to DD do
   begin
      for x2 := 1 to W do
      begin
```

```
            RK[x1,x2] := PROD[x1,x2];
        end;
    end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
    matprd(-1,RK,DD,DD,DD,W);
    for x1 := 1 to DD do
    begin
        for x2 := 1 to W do
        begin
            RL[x1,x2] := CONS[x1,x2];
        end;
    end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
    matprd(TD,-1,DD,W,DD,W);
    for x1 := 1 to DD do
    begin
        for x2 := 1 to W do
        begin
            RM[x1,x2] := CONS[x1,x2];
        end;
    end;
end;

if ((EE <> 0) and (A <> 0)) then
begin
    matmul(LJ,RM,EE,EE,EE,AA);
    for x1 := 1 to EE do
    begin
        for x2 := 1 to AA do
        begin
            RN[x1,x2] := PROD[x1,x2];
        end;
    end;
end;

if ((V <> 0) and (AA <> 0)) then
begin
    matmul(UC,RN,V,EE,EE,AA);
    for x1 := 1 to V do
    begin
        for x2 := 1 to AA do
        begin
            RO[x1,x2] := PROD[x1,x2];
        end;
    end;
end;

if ((DD <> 0) and (AA <> 0)) then
begin
```

```
    matmul(HM,RO,DD,V,V,AA);
    for x1 := 1 to V do
    begin
       for x2 := 1 to AA do
       begin
          RP[x1,x2] := PROD[x1,x2];
       end;
    end;
end;

if ((V <> 0) and (W <> 0)) then
begin
    matplu(RP,RL,V,DD,DD,W);
    for x1 := 1 to V do
    begin
       for x2 := 1 to W do
       begin
          RQ[x1,x2] := PLUS[x1,x2];
       end;
    end;
end;

if ((U <> 0) and (W <> 0)) then
begin
    matmul(H1,CJ,U,U,U,W);
    for x1 := 1 to U do
    begin
       for x2 := 1 to W do
       begin
          RS[x1,x2] := PROD[x1,x2];
       end;
    end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
    matmul(FE,RS,DD,U,U,W);
    for x1 := 1 to DD do
    begin
       for x2 := 1 to W do
       begin
          RU[x1,x2] := PROD[x1,x2];
       end;
    end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
    matprd(RT,-1,DD,W,DD,W);
    for x1 := 1 to DD do
    begin
       for x2 := 1 to W do
       begin
          RU[x1,x2] := CONS[x1,x2];
       end;
```

```
   end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
   matprd(TA,-1,DD,W,DD,W);
   for x1 := 1 to DD do
   begin
      for x2 := 1 to W do
      begin
         RV[x1,x2] := CONS[x1,x2];
      end;
   end;
end;

if ((EE <> 0) and (A <> 0)) then
begin
   matmul(CJ,RV,EE,EE,EE,AA);
   for x1 := 1 to EE do
   begin
      for x2 := 1 to AA do
      begin
         RU[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((V <> 0) and (AA <> 0)) then
begin
   matmul(UA,RW,V,EE,EE,AA);
   for x1 := 1 to V do
   begin
      for x2 := 1 to AA do
      begin
         RY[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((DD <> 0) and (AA <> 0)) then
begin
   matmul(FA,RX,DD,V,V,AA);
   for x1 := 1 to V do
   begin
      for x2 := 1 to AA do
      begin
         RY[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((V <> 0) and (W <> 0)) then
begin
   matplu(RS,RY,V,DD,DD,W);
   for x1 := 1 to V do
```

```
   begin
      for x2 := 1 to W do
      begin
         RZ[x1,x2] := PLUS[x1,x2];
      end;
   end;
end;

if ((U <> 0) and (W <> 0)) then
begin
   matmul(W7,LJ,U,U,U,W);
   for x1 := 1 to U do
   begin
      for x2 := 1 to W do
      begin
         R1[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
   matmul(FE,R1,DD,U,U,W);
   for x1 := 1 to DD do
   begin
      for x2 := 1 to W do
      begin
         R2[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
   matmin(FI,R2,DD,DD,DD,W);
   for x1 := 1 to DD do
   begin
      for x2 := 1 to W do
      begin
         R3[x1,x2] := MINU[x1,x2];
      end;
   end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
   matprd(TB,-1,DD,W,DD,W);
   for x1 := 1 to DD do
   begin
      for x2 := 1 to W do
      begin
         R4[x1,x2] := CONS[x1,x2];
      end;
   end;
end;
```

```
if ((EE <> 0) and (A <> 0)) then
begin
   matmul(CJ,R4,EE,EE,EE,AA);
   for x1 := 1 to EE do
   begin
      for x2 := 1 to AA do
      begin
         R5[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((V <> 0) and (AA <> 0)) then
begin
   matmul(UA,R5,V,EE,EE,AA);
   for x1 := 1 to V do
   begin
      for x2 := 1 to AA do
      begin
         R6[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((DD <> 0) and (AA <> 0)) then
begin
   matmul(FA,R6,DD,V,V,AA);
   for x1 := 1 to V do
   begin
      for x2 := 1 to AA do
      begin
         R7[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((V <> 0) and (W <> 0)) then
begin
   matplu(R3,R7,V,DD,DD,W);
   for x1 := 1 to V do
   begin
      for x2 := 1 to W do
      begin
         R8[x1,x2] := PLUS[x1,x2];
      end;
   end;
end;

{Formulation of C matrix)

   if ((BX <> 0) and (W <> 0)) then
   begin
      for C1 := 1 to DD do
      begin
```

```
            for C2 := 1 to W do
            begin
               C[C1,C2] := RH[C1,C2];
            end;
         end;
      end;

      C3 := W + 1;

      if (AA <> 0) then
      begin
         for C4 := 1 to DD do
         begin
            for C5 := C3 to BX do
            begin
               CA := C5 - DD;
               C[C4,C5] := RK[C4,CA];
            end;
         end;
      end;

      if ((V <> 0) and (W <> 0)) then
      begin
         for C6 := C3 to AX do
         begin
            for C7 := 1 to W do
            begin
               CC := C6 - DD;
               C[C6,C7] := RN[CC,C7];
            end;
         end;
      end;

      if (AA <> 0) then
      begin
         for C8 := C3 to AX do
         begin
            for C9 := C3 to BX do
            begin
               CD := C8 - DD;
               CF := C9 - DD;
               C[C8,C9] := RP[CD,CF];
            end;
         end;
      end;

      writeln(lst,'C');
      writeln(lst,'-');
      matpri(C,AX,BX);
      writeln(lst);

{ State matrix D }

      BX := W + AA;
```

```
if ((U <> 0) and (W <> 0)) then
begin
   matprd(TN,-1,U,U,U,W);
   for x1 := 1 to U do
   begin
      for x2 := 1 to W do
      begin
         SA[x1,x2] := CONS[x1,x2];
      end;
   end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
   matmul(LJ,SA,DD,U,U,W);
   for x1 := 1 to DD do
   begin
      for x2 := 1 to W do
      begin
         SB[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
   matmul(UC,SB,DD,DD,DD,W);
   for x1 := 1 to DD do
   begin
      for x2 := 1 to W do
      begin
         SC[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
   matprd(HM,SC,DD,W,DD,W);
   for x1 := 1 to DD do
   begin
      for x2 := 1 to W do
      begin
         SD[x1,x2] := CONS[x1,x2];
      end;
   end;
end;

if ((EE <> 0) and (A <> 0)) then
begin
   matmul(CJ,TF,EE,EE,EE,AA);
   for x1 := 1 to EE do
   begin
      for x2 := 1 to AA do
      begin
```

```
            SE[x1,x2] := PROD[x1,x2];
        end;
    end;
end;

if ((V <> 0) and (AA <> 0)) then
begin
    matmul(W9,SE,V,EE,EE,AA);
    for x1 := 1 to V do
    begin
        for x2 := 1 to AA do
        begin
            SF[x1,x2] := PROD[x1,x2];
        end;
    end;
end;

if ((DD <> 0) and (AA <> 0)) then
begin
    matmul(HI,SF,DD,V,V,AA);
    for x1 := 1 to V do
    begin
        for x2 := 1 to AA do
        begin
            SG[x1,x2] := PROD[x1,x2];
        end;
    end;
end;

if ((V <> 0) and (W <> 0)) then
begin
    matmin(HE,SG,V,DD,DD,W);
    for x1 := 1 to V do
    begin
        for x2 := 1 to W do
        begin
            SH[x1,x2] := MINU[x1,x2];
        end;
    end;
end;

if ((U <> 0) and (W <> 0)) then
begin
    matplu(SH,SD,U,U,U,W);
    for x1 := 1 to U do
    begin
        for x2 := 1 to W do
        begin
            SJ[x1,x2] := PROD[x1,x2];
        end;
    end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
```

```
    matmul(HI,W0,DD,U,U,W);
    for x1 := 1 to DD do
    begin
       for x2 := 1 to W do
       begin
          SK[x1,x2] := PROD[x1,x2];
       end;
    end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
    matmin(HA,SK,DD,DD,DD,W);
    for x1 := 1 to DD do
    begin
       for x2 := 1 to W do
       begin
          SL[x1,x2] := MINU[x1,x2];
       end;
    end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
    matmin(SL,SJ,DD,W,DD,W);
    for x1 := 1 to DD do
    begin
       for x2 := 1 to W do
       begin
          SM[x1,x2] := CONS[x1,x2];
       end;
    end;
end;

if ((U <> 0) and (W <> 0)) then
begin
    matprd(TQ,-1,U,U,U,W);
    for x1 := 1 to U do
    begin
       for x2 := 1 to W do
       begin
          SN[x1,x2] := CONS[x1,x2];
       end;
    end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
    matmul(LJ,SN,DD,U,U,W);
    for x1 := 1 to DD do
    begin
       for x2 := 1 to W do
       begin
          SO[x1,x2] := PROD[x1,x2];
       end;
```

```
    end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
    matmul(UC,SO,DD,DD,DD,W);
    for x1 := 1 to DD do
    begin
        for x2 := 1 to W do
        begin
            SP[x1,x2] := PROD[x1,x2];
        end;
    end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
    matmul(HM,SP,DD,W,DD,W);
    for x1 := 1 to DD do
    begin
        for x2 := 1 to W do
        begin
            SQ[x1,x2] := PROD[x1,x2];
        end;
    end;
end;

if ((EE <> 0) and (A <> 0)) then
begin
    matmul(LJ,TJ,EE,EE,EE,AA);
    for x1 := 1 to EE do
    begin
        for x2 := 1 to AA do
        begin
            SR[x1,x2] := PROD[x1,x2];
        end;
    end;
end;

if ((V <> 0) and (AA <> 0)) then
begin
    matmul(H6,SR,V,EE,EE,AA);
    for x1 := 1 to V do
    begin
        for x2 := 1 to AA do
        begin
            SS[x1,x2] := PROD[x1,x2];
        end;
    end;
end;

if ((DD <> 0) and (AA <> 0)) then
begin
    matmul(HI,SS,DD,V,V,AA);
    for x1 := 1 to V do
```

```
    begin
      for x2 := 1 to AA do
      begin
        ST[x1,x2] := PROD[x1,x2];
      end;
    end;
end;

if ((U <> 0) and (W <> 0)) then
begin
    matplu(ST,SQ,U,U,U,W);
    for x1 := 1 to U do
    begin
      for x2 := 1 to W do
      begin
        SU[x1,x2] := PLUS[x1,x2];
      end;
    end;
end;

if ((U <> 0) and (W <> 0)) then
begin
    matprd(TH,-1,U,U,U,W);
    for x1 := 1 to U do
    begin
      for x2 := 1 to W do
      begin
        SV[x1,x2] := CONS[x1,x2];
      end;
    end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
    matmul(UA,SV,DD,U,U,W);
    for x1 := 1 to DD do
    begin
      for x2 := 1 to W do
      begin
        SW[x1,x2] := PROD[x1,x2];
      end;
    end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
    matmul(FA,SW,DD,DD,DD,W);
    for x1 := 1 to DD do
    begin
      for x2 := 1 to W do
      begin
        SX[x1,x2] := PROD[x1,x2];
      end;
    end;
end;
```

```
if ((DD <> 0) and (W <> 0)) then
begin
   matmul(CJ,TF,DD,W,DD,W);
   for x1 := 1 to DD do
   begin
      for x2 := 1 to W do
      begin
         SY[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((EE <> 0) and (A <> 0)) then
begin
   matmul(FE,SY,EE,EE,EE,AA);
   for x1 := 1 to EE do
   begin
      for x2 := 1 to AA do
      begin
         SZ[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((V <> 0) and (AA <> 0)) then
begin
   matplu(SZ,SX,V,EE,EE,AA);
   for x1 := 1 to V do
   begin
      for x2 := 1 to AA do
      begin
         S1[x1,x2] := PLUS[x1,x2];
      end;
   end;
end;

if ((DD <> 0) and (AA <> 0)) then
begin
   matprd(FE,-1,DD,V,V,AA);
   for x1 := 1 to V do
   begin
      for x2 := 1 to AA do
      begin
         S2[x1,x2] := CONS[x1,x2];
      end;
   end;
end;

if ((V <> 0) and (W <> 0)) then
begin
   matmul(FE,H3,V,DD,DD,W);
   for x1 := 1 to V do
   begin
      for x2 := 1 to W do
```

```
      begin
         S3[x1,x2] := PROd[x1,x2];
      end;
   end;
end;

if ((U <> 0) and (W <> 0)) then
begin
   matplu(S3,S1,U,U,U,W);
   for x1 := 1 to U do
   begin
      for x2 := 1 to W do
      begin
         S4[x1,x2] := PLUS[x1,x2];
      end;
   end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
   matmul(TL,-1,DD,U,U,W);
   for x1 := 1 to DD do
   begin
      for x2 := 1 to W do
      begin
         S5[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
   matmul(CJ,S5,DD,DD,DD,W);
   for x1 := 1 to DD do
   begin
      for x2 := 1 to W do
      begin
         S6[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
   matmin(UA,S7,DD,W,DD,W);
   for x1 := 1 to DD do
   begin
      for x2 := 1 to W do
      begin
         S7[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((U <> 0) and (W <> 0)) then
```

```
begin
   matmul(HA,S7,U,U,U,W);
   for x1 := 1 to U do
   begin
      for x2 := 1 to W do
      begin
         S8[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
   matmul(LJ,TJ,DD,U,U,W);
   for x1 := 1 to DD do
   begin
      for x2 := 1 to W do
      begin
         S9[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
   matmul(W7,S9,DD,DD,DD,W);
   for x1 := 1 to DD do
   begin
      for x2 := 1 to W do
      begin
         S10[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((DD <> 0) and (W <> 0)) then
begin
   matmul(FE,S10,DD,W,DD,W);
   for x1 := 1 to DD do
   begin
      for x2 := 1 to W do
      begin
         S11[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((EE <> 0) and (A <> 0)) then
begin
   matmin(FI,S11,EE,EE,EE,AA);
   for x1 := 1 to EE do
   begin
      for x2 := 1 to AA do
      begin
         S12[x1,x2] := MINU[x1,x2];
```

```
      end;
   end;
end;

if ((V <> 0) and (AA <> 0)) then
begin
   matmul(FE,S12,V,EE,EE,AA);
   for x1 := 1 to V do
   begin
      for x2 := 1 to AA do
      begin
         S13[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((DD <> 0) and (AA <> 0)) then
begin
   matmin(FM,S13,DD,V,V,AA);
   for x1 := 1 to V do
   begin
      for x2 := 1 to AA do
      begin
         S14[x1,x2] := MINU[x1,x2];
      end;
   end;
end;

if ((U <> 0) and (W <> 0)) then
begin
   matprd(S15,-1,U,U,U,W);
   for x1 := 1 to U do
   begin
      for x2 := 1 to W do
      begin
         S16[x1,x2] := CONS[x1,x2];
      end;
   end;
end;

if ((DD <> 0) and (AA <> 0)) then
begin
   matplu(S8,S12,DD,V,V,AA);
   for x1 := 1 to V do
   begin
      for x2 := 1 to AA do
      begin
         S17[x1,x2] := PLUS[x1,x2];
      end;
   end;
end;

if ((U <> 0) and (W <> 0)) then
begin
   matplu(S17,S18,U,U,U,W);
```

```
      for x1 := 1 to U do
      begin
         for x2 := 1 to W do
         begin
            S18[x1,x2] := PLUS[x1,x2];
         end;
      end;
   end;

{Formulation of D matrix)

   if ((BX <> 0) and (W <> 0)) then
   begin
      for D1 := 1 to DD do
      begin
         for D2 := 1 to W do
         begin
            D[D1,D2] := SH[D1,D2];
         end;
      end;
   end;

   D3 := W + 1;

   if (AA <> 0) then
   begin
      for D4 := 1 to DD do
      begin
         for D5 := D3 to BX do
         begin
            DA := D5 - DD;
            D[D4,D5] := SK[D4,DA];
         end;
      end;
   end;

   if ((V <> 0) and (W <> 0)) then
   begin
      for D6 := D3 to AX do
      begin
         for D7 := 1 to W do
         begin
            DC := D6 - DD;
            D[D6,D7] := SN[DC,D7];
         end;
      end;
   end;

   if (AA <> 0) then
   begin
      for D8 := D3 to AX do
      begin
         for D9 := D3 to BX do
         begin
            DD := D8 - DD;
```

```
            DF := D9 - DD;
            D[D8,D9] := SP[DD,DF];
         end;
      end;
   end;

   writeln(lst,'C');
   writeln(lst,'-');
   matpri(C,AX,BX);
   writeln(lst);

{ State matrix E }

   BX := W + AA;

   if ((DD <> 0) and (AA <> 0)) then
   begin
      matmul(LJ,TJ,DD,V,V,AA);
      for x1 := 1 to V do
      begin
         for x2 := 1 to AA do
         begin
            P1[x1,x2] := PROD[x1,x2];
         end;
      end;
   end;

   if ((U <> 0) and (W <> 0)) then
   begin
      matmul(UC,P1,U,U,U,W);
      for x1 := 1 to U do
      begin
         for x2 := 1 to W do
         begin
            P2[x1,x2] := PROD[x1,x2];
         end;
      end;
   end;

   if ((DD <> 0) and (AA <> 0)) then
   begin
      matmul(HM,P2,DD,V,V,AA);
      for x1 := 1 to V do
      begin
         for x2 := 1 to AA do
         begin
            P3[x1,x2] := PROD[x1,x2];
         end;
      end;
   end;

   if ((U <> 0) and (W <> 0)) then
   begin
      matprd(HM,-1,U,U,U,W);
      for x1 := 1 to U do
```

```
    begin
      for x2 := 1 to W do
      begin
        P4[x1,x2] := CONS[x1,x2];
      end;
    end;
end;

if ((DD <> 0) and (AA <> 0)) then
begin
    matmul(P4,TI,DD,V,V,AA);
    for x1 := 1 to V do
    begin
      for x2 := 1 to AA do
      begin
        P5[x1,x2] := PROD[x1,x2];
      end;
    end;
end;

if ((U <> 0) and (W <> 0)) then
begin
    matplu(P5,P3,U,U,U,W);
    for x1 := 1 to U do
    begin
      for x2 := 1 to W do
      begin
        P6[x1,x2] := PLUS[x1,x2];
      end;
    end;
end;

if ((DD <> 0) and (AA <> 0)) then
begin
    matmul(CJ,TF,DD,V,V,AA);
    for x1 := 1 to V do
    begin
      for x2 := 1 to AA do
      begin
        P7[x1,x2] := PROD[x1,x2];
      end;
    end;
end;

if ((U <> 0) and (W <> 0)) then
begin
    matmul(UA,P7,U,U,U,W);
    for x1 := 1 to U do
    begin
      for x2 := 1 to W do
      begin
        P8[x1,x2] := PROD[x1,x2];
      end;
    end;
end;
```

```
if ((DD <> 0) and (AA <> 0)) then
begin
   matmul(FA,P9,DD,V,V,AA);
   for x1 := 1 to V do
   begin
      for x2 := 1 to AA do
      begin
         P10[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((U <> 0) and (W <> 0)) then
begin
   matprd(FA,-1,U,U,U,W);
   for x1 := 1 to U do
   begin
      for x2 := 1 to W do
      begin
         P10[x1,x2] := CONS[x1,x2];
      end;
   end;
end;

if ((DD <> 0) and (AA <> 0)) then
begin
   matmul(P10,TE,DD,V,V,AA);
   for x1 := 1 to V do
   begin
      for x2 := 1 to AA do
      begin
         P11[x1,x2] := PROD[x1,x2];
      end;
   end;
end;

if ((U <> 0) and (W <> 0)) then
begin
   matplu(P11,P9,U,U,U,W);
   for x1 := 1 to U do
   begin
      for x2 := 1 to W do
      begin
         P12[x1,x2] := PLUS[x1,x2];
      end;
   end;
end;

{Formulation of E matrix)

   if ((BX <> 0) and (W <> 0)) then
   begin
      for E1 := 1 to DD do
      begin
```

```
            for E2 := 1 to W do
            begin
               E[E1,E2] := 0;
            end;
         end;
      end;

      E3 := W + 1;

      if (AA <> 0) then
      begin
         for E4 := 1 to DD do
         begin
            for E5 := E3 to BX do
            begin
               EA := E5 - DD;
               E[E4,E5] := P6[E4,EA];
            end;
         end;
      end;

      if ((V <> 0) and (W <> 0)) then
      begin
         for E6 := E3 to AX do
         begin
            for E7 := 1 to W do
            begin
               EC := E6 - DD;
               E[E6,E7] := P12[EC,E7];
            end;
         end;
      end;

      if (AA <> 0) then
      begin
         for E8 := E3 to AX do
         begin
            for E9 := E3 to BX do
            begin
               ED := E8 - DD;
               EF := E9 - DD;
               E[E8,E9] := 0;
            end;
         end;
      end;

      writeln(lst,'E');
      writeln(lst,'-');
      matpri(E,AX,BX);
      writeln(lst);

end;

begin      { Start of code section for procedure STATEM }
```

```
        elem;
        inv1;
        int1;
        inv2;
        hybr;
        stat;

end;

END.
```

```
{$B+}   {Boolean complete evaluation on}
{$S+}   {Stack checking on}
{$I+}   {I/O checking on}
{$N+}   {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT GCOPPT;        {Version 1      Created on 14th June 1991}

{$R+}   {Compiler range checking for arrays, etc.}
{$F+}   {Far call compiler directive}
{$O+}   {Overlay compiler directive}


{
 GCOPPT is an overlaid unit of GCAP03, a Generalised Circuit Analysis Program
 for p lsed power.  This is version 1 of the unit which is used to
 calculate the operating point of a circuit containing nonlinear elements
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Printer,
  GcInit,
  GcStat,
  GcTemp;

VAR oppt                        :boolean;
      tolerance, value          :array[1..3] of real;

procedure inittv;
procedure newton_raphson;
procedure dcoppt;

Implementation

{------------------------------------------------------------------------}

{Now declare the procedures and functions}

PROCEDURE INITTV;

{
 Procedure to calculate the initial non-linear and time-varying conditions
}

var   number, index                             :integer;

begin               {Start of code section for INITTV}
```

```
      oppt := false;
      index := 0;

      for number := 1 to num1 do
      begin
         if COMP7[number,4] <> 'LIN' then
         begin                    {Set the initial non-linear conditions}
            oppt := true;
            index := index + 1;
            COMP8[number,1] := 0.5*COMP8[number,1];
            value[index] := 0.5*COMP8[number,1];
            tolerance[index] := 0.05*COMP8[number,1];       {5% tolerance}
         end;
         if COMP7[number,5] <> 'INV' then
         begin                    {Set the initial time-varying conditions}
            oppt := true;
            index := index + 1;
            COMP8[number,1] := COMP8[number,4];
            value[index] := COMP8[number,4];
         end;
      end;
end;              {End of code section for INITTV}

PROCEDURE NEWTON_RAPHSON;

{
 Newton-Raphson algorithm to iterate the simulation for a DC analysis
}

var   number, num, nom                          :integer;
      DT1, DT2, DT3                             :real;

begin             {Start of code section for NEWTON-RAPHSON}

         for num := 1 to num1 do
         begin
            for nom := 1 to num1 do
            begin
            if inc_time < A[num,nom]/5 then  {Set time increment to one-fifth
            inc_time :=  A[num,nom]/5;        of the smallest time constant
            end;                              inherent in the A-matrix}
         end;

         DT1 := inc_time*inc_time/2;
         DT2 := inc_time*inc_time*inc_time/6;
         DT3 := inc_time*inc_time*inc_time*inc_time/24;

         rungek;

end;              {End of code section for NEWTON-RAPHSON}


PROCEDURE DCOPPT;
```

```
{
DC analysis to calculate the DC operating point of non-linear circuit
elements (initial non-linear conditions) by means of a Newton-Raphson
algorithm. Inductors are replaced by 1 milliohm resistors, and all
time-varying sources, switches and capacitances are removed. The
fundamental cutset matrix Q is re-calculated. The operational values
of non-linear components are set to 50% of the full value. A
Newton-Raphson algorithm is invoked to estimate a new value. If
convergence to a 5% tolerance (set in the unit INITTV) is not achieved
within 10 iterations, then the process is repeated with a revised
initial estimate which is 50% above the previous value
}

var   number, num, nom,
      index                                   :integer;
      converge                                :array[1..3] of boolean;

begin              {Start of code section for DCOPPT}

    if oppt then
    begin
       for nom := 1 to 3 do
          converge[nom] := false;
       repeat
       inittv;
       for number := 1 to num1 do
       begin
          if COMP7[number,1] = 'L' then
             COMP8[number,1] := 1E-3
          else if COMP7[number,1] = 'C' then
             COMP8[number,1] := 1.0
          else if ((COMP7[number,1] = 'V') and
           (COMP7[number,4] = 'INV')) then
             COMP8[number,1] := 1E-3;
       end;

          num := 0;
          repeat
          num := num + 1;
          statem;
          newton_raphson;
          index := 0;
          for nom := 1 to num1 do
          begin
             if COMP7[num,4] <> 'LIN' then
             begin
                index := index + 1;
                if sqrt(sqr(COMP8[nom,1] - value[index])) <= tolerance[index] then
                begin
                   converge[nom] := true;
                end;
                if (num = 10) and (converge[nom] := false) then
                begin
                   COMP8[nom,1] := 0.5*COMP8[nom,1];
                   exit;
```

```
                end;
              end;
            end;
            until num >= 10;

        until (converge[1] and converge[2] and converge[3]);
    end;
end;                {End of code section for DCOPPT}

END.
```

```
{$B+}   {Boolean complete evaluation on}
{$S+}   {Stack checking on}
{$I+}   {I/O checking on}
{$N+}   {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT GCDC;          {Version 1     Created on 7th May 1991}

{$R+}   {Compiler range checking for arrays, etc.}
{$F+}   {Far call compiler directive}
{$O+}   {Overlay compiler directive}

{
 GCDC is an overlaid unit of GCAP03, a Generalised Circuit Analysis Program
 for pulsed power.  This is version 1 of the unit which performs a non-linear
 DC analysis prior to AC, transient or frequency analyses on the circuit to
 be analysed.
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Printer,
  GcInit,
  GcTemp,
  GcStat,
  GcOut2;

procedure nonlindc;

Implementation

{-------------------------------------------------------------------------}

{Now declare the procedures and functions}


PROCEDURE NONLINDC;

{
 This is the main non-linear DC analysis procedure, which performs a DC
 analysis on the circuit to produce a DC transfer curve (low voltage
 excitation)
}

var   num, number, nom,
      incr                              :integer;
      initial_value,
      input, output,
```

```
      DT1, DT2, DT3                    :real;

begin

     number := 0;

     for num := 1 to num1 do
     begin
        if COMP7[num,1] = 'V' then
        begin
           initial_value := COMP8[num,1];
           for number := 1 to 10 do
           begin
              COMP8[num,1] := (0.1*number)*initial_value;

              for num := 1 to num1 do
              begin
                 for nom := 1 to num1 do
                 begin
                 if inc_time < A[num,nom]/5 then  {Set time increment to one-fifth
                 inc_time :=  A[num,nom]/5;        of the smallest time constant
                 end;                              inherent in the A-matrix}
              end;

              DT1 := inc_time*inc_time/2;
              DT2 := inc_time*inc_time*inc_time/6;
              DT3 := inc_time*inc_time*inc_time*inc_time/24;

              incr := 0;
              incr := incr + 1;
              repeat
                 rungek;
              until (incr = 10);
              input := COMP8[num,1];
              output := VOLTAGE[num,incr];
            end;
           end;
        end;
     end;

     twodgraph;

end;    {Of code section of NONLINDC}


END.
```

```
{$B+}   {Boolean complete evaluation on}
{$S+}   {Stack checking on}
{$I+}   {I/O checking on}
{$N+}   {Numeric coprocessor}
{$M 65500,16384,655360}{Turbo 3 default stack and heap}


UNIT GCAC;          {Version 1     Created on 7th May 1991}

{$R+}   {Compiler range checking for arrays, etc.}
{$F+}   {Far call compiler directive}
{$O+}   {Overlay compiler directive}

{
 GCAC is an overlaid unit of GCAP03, a Generalised Circuit Analysis Program
 for pulsed power.  This is version 1 of the unit which performs a linear
 AC analysis on the circuit.
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Printer,
  GcInit,
  GcTemp,
  GcStat,
  GcOut2;

procedure linearac;
procedure actran;
procedure bode;
procedure gainm;
procedure stabm;

Implementation

{-------------------------------------------------------------------------}

{Now declare the procedures and functions}

PROCEDURE ACTRAN;

{
 This is the main linear AC analysis procedure, which performs an AC
 analysis on the circuit to produce an AC transfer curve (sinusoidal
 voltage excitation)
}

var   num, number, nom,
      incr                                  :integer;
```

```
      initial_value,
      input, output,
      DT1, DT2, DT3,
      inc_freq                              :real;

begin

     number := 0;

     for num := 1 to num1 do
     begin
        if COMP7[num,1] = 'V' then
        begin
           initial_value := COMP8[num,1];
           for number := 1 to 10 do
           begin
              COMP8[num,1] := initial_value * sin(number*pi/20);

              DT1 := inc_time*inc_time/2;
              DT2 := inc_time*inc_time*inc_time/6;
              DT3 := inc_time*inc_time*inc_time*inc_time/24;

              incr := 0;
              incr := incr + 1;
              repeat
                 rungek;
              until (incr = 10);
              input := COMP8[num,1];
              output := VOLTAGE[num,incr];
           end;
        end;
     end;

     twodgraph;

end;  {Of code section of ACTRAN}

PROCEDURE BODE;

{
 Procedure to produce a Bode plot
}

var   num, number, nom,
      incr                              :integer;
      initial_value,
      input, output,
      DT1, DT2, DT3,
      inc_freq                          :real;

begin

     number := 0;

     for num := 1 to num1 do
```

```
    begin
      if COMP7[num,1] = 'V' then
      begin
        initial_value := COMP8[num,1];
        for number := 1 to 10 do
        begin
          COMP8[num,1] := initial_value * sin(number*pi/20);

          DT1 := inc_time*inc_time/2;
          DT2 := inc_time*inc_time*inc_time/6;
          DT3 := inc_time*inc_time*inc_time*inc_time/24;

          incr := 0;
          incr := incr + 1;
          repeat
            rungek;
          until (incr = 10);
          input := COMP8[num,1];
          output := VOLTAGE[num,incr];
        end;
      end;
    end;

    twodgraph;

end;   {Of code section of BODE}

PROCEDURE GAINM;

{
 Procedure to calculate the gain margin of the circuit:
    20*log[G(jw)]; 180 degrees  (dB)
 and the phase margin of the circuit:
    20*log[G(jw)] = 0  (degrees)
}

var   num, number                              :integer;

      maximum,
      gainm, phasem                            :real;

begin  {Start of code section of GAINM}

    for num := 1 to num1 do
    begin
      if COMP7[num,1] = 'V' then
      begin
        for number := 2 to 10 do
        begin
          if VOLTAGE[num,number] > VOLTAGE[num,number-1] then
          begin
            maximum := VOLTAGE[num,number];
            gainm := VOLTAGE[num,number];
            phasem := VOLTAGE[num,number];
            writeln(lst,'GAIN MARGIN = ',gainm:11);
```

```
                writeln(lst,'PHASE MARGIN = ',phasem:11);
              end;
            end;
          end;
        end;

end;    {Of code section of GAINM}


PROCEDURE STABM;

{
 Procedure to calculate the stability margin of the circuit:
    20*log[G(jw)] < 0   (dB)
}

var    num, number                                    :integer;

       minimum, negative,
       stabm                                          :real;

begin  {Start of code section of STABM}

      for num := 1 to num1 do
      begin
         if COMP7[num,1] = 'V' then
         begin
            for number := 2 to 10 do
            begin
               if (VOLTAGE[num,number] < 0) and
               (VOLTAGE[num,number] < VOLTAGE[num,number]) then
               begin
                  minimum := VOLTAGE[num,number];
                  stabm := minimum;
                  writeln(lst,'STABILITY MARGIN = ',stabm:11);
               end;
            end;
         end;
      end;

end;    {Of code section of STABM}


PROCEDURE LINEARAC;

{
 This is the main linear AC analysis procedure, which performs an AC
 analysis on the circuit to produce an AC transfer curve
}

var    nem, nem1                                      :integer;

begin             {Start of code section for LINEARAC}

      actran;
```

```
      for nem := 1 to num2 do
      begin
         if ANLY1[nem,1] = 'BODE' then
         begin
            bode;
            for nem1 := 1 to num2 do
            begin
               if ANLY1[nem,1] = 'GAINM' then
               begin
                  gainm;
               end
               else if ANLY1[nem,1] = 'STABM' then
               begin
                  stabm;
               end;
            end;
         end;
      end;

end;   {Of code section of LINEARAC}


END.
```

```
{$B+}   {Boolean complete evaluation on}
{$S+}   {Stack checking on}
{$I+}   {I/O checking on}
{$N+}   {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT GCTEMP;        {Version 1      Created on 7th May 1991}

{$R+}   {Compiler range checking for arrays, etc.}
{$F+}   {Far call compiler directive}
{$O+}   {Overlay compiler directive}

{
 SRTEMP0 is an overlaid unit of GCAP03, a Generalised Circuit Analysis Program
 for pulsed power.  This is version 1 of the unit which performs a transient
 analysis on the circuit.
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Printer,
  GcMat,
  GcInit,
  GcStat;

VAR time                                        :real;

CONST   max_num_AX = 10;
        max_num_BX = 10;
        max_num_CX = 10;
        max_num_DX = 10;
        max_num_EX = 10;

procedure nonlintra;
procedure rungek;
procedure linear;

Implementation

{------------------------------------------------------------------------}

{Now declare the procedures and functions}

PROCEDURE RUNGEK;

{
 Fourth-order Runge-Kutta algorithm for the solution of the state
 equation
```

```
}

var   A10, A11, A12, A13,
      A14, A15, A16, A17,
      A18, A19, A20, A21,
      A22, A23, A24,
      B11, B12, B13, B14,
      B15, B18, B19, B20,
      B25, B26, B27,
      B28, B29, B30,
      ZIR, ZSR, tot_resp      :array[1..max_num_AX,1..max_num_AX] of real;

      X, X11,
      YA, YB, YO,
      A25,
      B16, B17, B24,
      B31, B32                :array[1..max_num_AX,1..max_num_BX] of real;

      x1, x2,
      temp                    :integer;

      DT1 := inc_time*inc_time/2;
      DT2 := inc_time*inc_time*inc_time/6;
      DT3 := inc_time*inc_time*inc_time*inc_time/24;

begin   {Start of code section for RUNGEK}

      matidn(A10,AX,AX,AX,AX);
      for x1 := 1 to AX do
      begin
         for x2 := 1 to AX do
         begin
            A10[x1,x2] := IDEN[x1,x2];
         end;
      end;

      matprd(A,inc_time,AX,AX,AX,AX);
      for x1 := 1 to AX do
      begin
         for x2 := 1 to AX do
         begin
            A11[x1,x2] := MAT13[x1,x2];
         end;
      end;

      matmul(A,A,AX,AX,AX,AX);
      for x1 := 1 to AX do
      begin
         for x2 := 1 to AX do
         begin
            A12[x1,x2] := PROD[x1,x2];
         end;
      end;

      matmul(A,A12,AX,AX,AX,AX);
```

```
for x1 := 1 to AX do
begin
   for x2 := 1 to AX do
   begin
      A13[x1,x2] := PROD[x1,x2];
   end;
end;

matmul(A,A13,AX,AX,AX,AX);
for x1 := 1 to AX do
begin
   for x2 := 1 to AX do
   begin
      A14[x1,x2] := PROD[x1,x2];
   end;
end;

matprd(A12,DT1,AX,AX,AX,AX);
for x1 := 1 to AX do
begin
   for x2 := 1 to AX do
   begin
      A15[x1,x2] := MAT13[x1,x2];
   end;
end;

matprd(A13,DT2,AX,AX,AX,AX);
for x1 := 1 to AX do
begin
   for x2 := 1 to AX do
   begin
      A16[x1,x2] := MAT13[x1,x2];
   end;
end;

matprd(A14,DT3,AX,AX,AX,AX);
for x1 := 1 to AX do
begin
   for x2 := 1 to AX do
   begin
      A17[x1,x2] := MAT13[x1,x2];
   end;
end;

matplu(A10,A11,AX,AX,AX,AX);
for x1 := 1 to AX do
begin
   for x2 := 1 to AX do
   begin
      A18[x1,x2] := PLUS[x1,x2];
   end;
end;

matplu(A18,A15,AX,AX,AX,AX);
for x1 := 1 to AX do
```

```
begin
   for x2 := 1 to AX do
   begin
      A19[x1,x2] := PLUS[x1,x2];
   end;
end;

matplu(A19,A16,AX,AX,AX,AX);
for x1 := 1 to AX do
begin
   for x2 := 1 to AX do
   begin
      A20[x1,x2] := PLUS[x1,x2];
   end;
end;

matplu(A20,A17,AX,AX,AX,AX);
for x1 := 1 to AX do
begin
   for x2 := 1 to AX do
   begin
      A21[x1,x2] := PLUS[x1,x2];
   end;
end;

matzer(X,AX,AX,AX,AX);
for x1 := 1 to AX do
begin
   for x2 := 1 to AX do
   begin
      X[x1,x2] := ZERO[x1,x2];
   end;
end;

matmul(A21,X,AX,AX,AX,1);
for x1 := 1 to AX do
begin
   x2 := 1;
   begin
      A25[x1,x2] := PROD[x1,x2];
   end;
end;

matprd(A10,1.833,AX,AX,AX,AX);
for x1 := 1 to AX do
begin
   for x2 := 1 to AX do
   begin
      B11[x1,x2] := MAT13[x1,x2];
   end;
end;

matprd(A11,0.5,AX,AX,AX,AX);
for x1 := 1 to AX do
begin
```

```
   for x2 := 1 to AX do
   begin
      B12[x1,x2] := MAT13[x1,x2];
   end;
end;

matprd(A15,0.33,AX,AX,AX,AX);
for x1 := 1 to AX do
begin
   for x2 := 1 to AX do
   begin
      B13[x1,x2] := MAT13[x1,x2];
   end;
end;

matprd(A16,0.25,AX,AX,AX,AX);
for x1 := 1 to AX do
begin
   for x2 := 1 to AX do
   begin
      B14[x1,x2] := MAT13[x1,x2];
   end;
end;

matplu(B11,B12,AX,AX,AX,AX);
for x1 := 1 to AX do
begin
   for x2 := 1 to AX do
   begin
      B15[x1,x2] := PLUS[x1,x2];
   end;
end;

matplu(B15,B13,AX,AX,AX,AX);
for x1 := 1 to AX do
begin
   for x2 := 1 to AX do
   begin
      B18[x1,x2] := PLUS[x1,x2];
   end;
end;

matplu(B18,B14,AX,AX,AX,AX);
for x1 := 1 to AX do
begin
   for x2 := 1 to AX do
   begin
      B19[x1,x2] := PLUS[x1,x2];
   end;
end;

matmul(B,U,AX,BX,BX,1);
for x1 := 1 to AX do
begin
   x2 := 1;
```

```
    begin
       B16[x1,x2] := PROD[x1,x2];
    end;
end;

matmul(B15,B16,AX,AX,AX,1);
for x1 := 1 to AX do
begin
    x2 := 1;
    begin
       B17[x1,x2] := PROD[x1,x2];
    end;
end;

writeln(lst,'A25');
writeln(lst,'---');
matpri(A25,AX,1);
writeln(lst);

writeln(lst,'B16');
writeln(lst,'---');
matpri(B16,AX,1);
writeln(lst);

writeln(lst,'B17');
writeln(lst,'---');
matpri(B17,BX,1);
writeln(lst);

matplu(A25,B17,AX,1,AX,1);
for x1 := 1 to AX do
begin
    x2 := 1;
    begin
       A25[x1,x2] := PLUS[x1,x2];
    end;
end;

matmch(X11,AX,1,AX,1);
for x1 := 1 to AX do
begin
    x2 := 1;
    begin
       X[x1,x2] := MAT14[x1,x2];
    end;
end;

matmul(A21,X,AX,AX,AX,1);
for x1 := 1 to AX do
begin
    x2 := 1;
    begin
       A21[x1,x2] := PROD[x1,x2];
    end;
end;
```

```
      matplu(A25,B17,AX,1,AX,1);
      for x1 := 1 to AX do
      begin
         x2 := 1;
         begin
            X11[x1,x2] := PLUS[x1,x2];
         end;
      end;

      matmch(B17,AX,1,AX,1);
      for x1 := 1 to AX do
      begin
         x2 := 1;
         begin
            ZSR[x1,x2] := MAT14[x1,x2];
         end;
      end;

      matmch(A25,AX,1,AX,1);
      for x1 := 1 to AX do
      begin
         x2 := 1;
         begin
            ZIR[x1,x2] := MAT14[x1,x2];
         end;
      end;

      matplu(ZIR,ZSR,AX,1,AX,1);
      for x1 := 1 to AX do
      begin
         x2 := 1;
         begin
            tot_resp[x1,x2] := PLUS[x1,x2];
         end;
      end;

end;    {Of code section of RUNGEK}

PROCEDURE LINEAR;

{
 Linear solution of the input-state-output state equation
}

var   C10                  :array[1..max_num_CX,1..max_num_CX] of real;

      X                    :array[1..max_num_AX,1..max_num_BX] of real;

      D10, E10,
      Y11, C25             :array[1..max_num_AX,1] of real;

      x1, x2,
      temp                 :integer;
```

```
      DT1 := inc_time;

begin   {Start of code section for LINEAR}

      matmul(C,X,CX,CX,CX,1);
      for x1 := 1 to CX do
      begin
         for x2 := 1 to CX do
         begin
            C10[x1,x2] := PROD[x1,x2];
         end;
      end;

      matmul(D,U,AX,BX,BX,1);
      for x1 := 1 to AX do
      begin
         x2 := 1;
         begin
            D10[x1,x2] := PROD[x1,x2];
         end;
      end;

      matmul(E,U,AX,BX,BX,1);
      for x1 := 1 to AX do
      begin
         x2 := 1;
         begin
            E10[x1,x2] := PROD[x1,x2];
         end;
      end;

      matplu(D10,E10,AX,1,AX,B1);
      for x1 := 1 to AX do
      begin
         x2 := 1;
         begin
            Y11[x1,x2] := PLUS[x1,x2];
         end;
      end;

      matplu(C10,Y11,AX,1,AX,1);
      for x1 := 1 to AX do
      begin
         x2 := 1;
         begin
            C25[x1,x2] := PLUS[x1,x2];
         end;
      end;

      writeln(lst,'C25');
      writeln(lst,'---');
      matpri(B25,AX,1);
      writeln(lst);

      writeln(lst,'C10');
```

```
      writeln(lst,'---');
      matpri(C10,AX,1);
      writeln(lst);

      writeln(lst,'D10');
      writeln(lst,'---');
      matpri(D10,BX,1);
      writeln(lst);

      writeln(lst,'E10');
      writeln(lst,'---');
      matpri(E10,BX,1);
      writeln(lst);

end;     {Of code section of LINEAR}

PROCEDURE NONLINTRA;

{
 This is the main temporal analysis procedure, which performs a transient
 analysis on the circuit
}

var   DT1, DT2, DT3,
      U, Vo,
      T, TIC                                    :real;
      num, number, nom                          :integer;

PROCEDURE TIMESET;

begin   {Start of code section for TIMESET}

     writeln(lst,'TIMESET');                    {Superceded by NONLINTRA}

end;     {Of code section of TIMESET}

PROCEDURE RESPON;

var   ch                                  :string[1];

begin   {Start of code section for RESPON}

   clrscr;
   gotoxy(1,2);
   writeln('       RESPONSE TO SPECIFIC INPUT FUNCTIONS');
   writeln('       ----------------------------------');
   writeln;
   writeln('       1. Unit step input');
   writeln('       2. Step input of specified magnitude');
   writeln('       3. Delta function');
   writeln('       4. Pulse');
   writeln('       5. Sinusoidal');
   writeln('       6. Square wave');
   writeln('       7. Exponential');
   writeln;
```

```
writeln('Hit any number to continue');
readln(ch);

if (ch = '1') then
begin
   U := 1;
end;

if (ch = '2') then
begin
   clrscr;
   gotoxy(1,2);
   writeln('STEP INPUT OF SPECIFIED MAGNITUDE');
   writeln('--------------------------------');
   writeln;
   writeln('Type in value of voltage source, Vv');
   readln(U);
end;

if (ch = '3') then
begin
   clrscr;
   gotoxy(1,2);
   writeln('DELTA FUNCTION VOLTAGE SOURCE');
   writeln('----------------------------');
   writeln;
   writeln('Type in value of voltage source, Vv');
   readln(U);
end;

if (ch = '4') then
begin
   clrscr;
   gotoxy(1,2);
   writeln('PULSE VOLTAGE SOURCE');
   writeln('--------------------');
   writeln;
   writeln('Type in value of voltage source, Vv');
   readln(U);
end;

if (ch = '5') then
begin
   clrscr;
   gotoxy(1,2);
   writeln('SINUSOIDAL VOLTAGE SOURCE');
   writeln('------------------------');
   writeln;
   writeln('Type in value of voltage source, Vv');
   readln(U);
end;

if (ch = '6') then
begin
   clrscr;
```

```
      gotoxy(1,2);
      writeln('SQUARE WAVE VOLTAGE SOURCE');
      writeln('-------------------------');
      writeln;
      writeln('Type in value of voltage source, Vv');
      readln(U);
   end;

   if (ch = '7') then
   begin
      clrscr;
      gotoxy(1,2);
      writeln('EXPONENTIAL VOLTAGE SOURCE');
      writeln('-------------------------');
      writeln;
      writeln('This input is of the form Vo*exp(-t/T)');
      writeln;
      writeln('Type in value of voltage source, Vv');
      readln(Vo);
      writeln('Type in time constant, T (Note: T is negative for an');
      writeln('exponentially growing source');
      readln(TIC);

      U := Vo*exp(-T/TIC);

   end;

end;    {Of code section of RESPON}


PROCEDURE IMPULS;

var   rise_time, fall_time, overshoot,
      rise_0, rise_10, rise_50, rise_90, rise_100,
      fall_0, fall_10, fall_50, fall_90, fall_100,
      FW_10, FWHM, FW_90,
      MAX, MAX_CURR,
      MIN, MIN_CURR                              :real;

      number, MAX_TIME, MIN_TIME                 :integer;


begin   {Start of code section for IMPULS}

         for number := 1 to num1 do
         begin
            CURRENT[number,0] := 0.0;
         end;


         for number := 1 to temp do
         begin
            if CURRENT[number,temp] > CURRENT[number,temp-1] then
            MAX_CURR := CURRENT[number,temp];
            MAX_TIME := number;
```

```
    end;

    for number := 1 to MAX_TIME do
    begin
       if CURRENT[number,temp] = 0.1*MAX then
       begin
          rise_10 := number;
       end;
    end;

    for number := 1 to MAX_TIME do
    begin
       if CURRENT[number,temp] = 0.5*MAX then
       begin
          rise_50 := number;
       end;
    end;

    for number := 1 to MAX_TIME do
    begin
       if CURRENT[number,temp] = 0.9*MAX then
       begin
          rise_90 := number;
       end;
    end;

    for number := MAX_TIME to temp do
    begin
       if CURRENT[number,temp] = 0.5*MAX then
       begin
          fall_50 := number;
       end;
    end;

    for number := MAX_TIME to temp do
    begin
       if CURRENT[number,temp] = 0.9*MAX then
       begin
          fall_90 := number;
       end;
    end;

    for number := MAX_TIME to temp do
    begin
       if CURRENT[number,temp] = 0.1*MAX then
       begin
          fall_10 := number;
       end;
    end;

FW_10 := (fall_10 - rise_10);
FWHM := (fall_50 - rise_50);
FW_90 := (fall_90 - rise_90);

writeln(lst,'Current rise-time (10-90%) = ',(rise_90-rise_10):11);
```

```
      writeln(lst,'Current fall-time (90-10%) = ',(fall_10-fall_90):11);
      writeln(lst,'Current pulse duration (FWHM) = ',(fall_50-rise_50):11);

end;    {Of code section of IMPULS}


PROCEDURE TEMPRE;

{
 Procedure to study the effect of temperature variation on system behaviour
 }

var   nem                              :integer;
      temperature                      :real;

begin   {Start of code section for TEMPRE}

          for nem := 1 to num1 do
          begin
             temperature := 300;
             if COMP8[number,4] <> 0 then
             repeat
                temperature := temperature + 25; {25 degree temperature variation}
                rungek
             until temperature > 600;
          end;

end;    {Of code section of TEMPRE}


PROCEDURE SENSE;

{
 Procedure to study the effect of individual parameter variation on system
 behaviour
 }

var    nem                                    :integer;

begin   {Start of code section for SENSE}

          for nem := 1 to num1 do
          begin
             COMP8[nem,1] := 1.05*COMP8[nem,1];      {5% tolerance}
             rungek;
          end;

end;    {Of code section of SENSE}

begin       {Start of code section for NONLINTRA}

{       header;       }          {Printout and display values in use}

          for num := 1 to num1 do
          begin
```

```
            for nom := 1 to num1 do
            begin
            if inc_time < A[num,nom]/5 then  {Set time increment to one-fifth
            inc_time :=  A[num,nom]/5;        of the smallest time constant
            end;                              inherent in the A-matrix}
         end;

         for number := 1 to num1 do
         begin
            if COMP7[number,1] = 'V' then
            U := COMP8[number,1];
         end;

         timeset;
         respon;
         time := 0;
         repeat
            rungek;
            linear;
         temp := temp + 1;
         input1[temp] := time;
         output1temp := VOLTAGE[num,temp];
         output2[temp] := CURRENT[num,temp];
         output3[temp] := VOLTAGE[num,temp]*CURRENT[num,temp]
         until (time >= sim_time);
         impuls;

         for number := 1 to num2 do
         begin
            if ANLY1[number,1] = 'TEMPRE' then
            tempre;
         end;

         for number := 1 to num2 do
         begin
            if ANLY1[number,1] = 'SENSE' then
            sense;
         end;
   end;

end;   {Of code section of NONLINTRA}

END.
```

```
{$B+}   {Boolean complete evaluation on}
{$S+}   {Stack checking on}
{$I+}   {I/O checking on}
{$N+}   {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT GCFREQ;        {Version 1      Created on 7th May 1991}

{$R+}   {Compiler range checking for arrays, etc.}
{$F+}   {Far call compiler directive}
{$O+}   {Overlay compiler directive}

{
 GCFREQ is an overlaid unit of GCAP03, a Generalised Circuit Analysis
 Program for pulsed power.  This is version 1 of the unit which
 performs a frequency analysis of the circuit to be analysed.
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Printer,
  GcInit,
  GcTemp,
  GcStat,
  GcOut2;

procedure frequency;

Implementation

{------------------------------------------------------------------------}

{Now declare the procedures and functions}


PROCEDURE FREQUENCY;

{
 This is the main frequency analysis procedure, which performs a frequency
 analysis on the circuit
}

var   nem, nem1, nom,
      incr                    :integer;
      initial_value,
      input, output,
      DT1, DT2, DT3,
      inc_freq                :real;
```

```
      Fs                          :array[1..max_num_AX,1..max_num_BX] of real;

      G1, G2, G3, G4,
      F1, F2, F3, F4,
      I1, I2, I3, I4,
      I5, I6, I7, I8              :array[1..max_num_AX,1..max_num_AX] of real;

procedure transf;

{
 Procedure to calculate the transfer function matrix H(s)
}

var   num, number                 :integer;

      H10, H11, H12               :array[1..max_num_AX,1..max_num_AX] of real;

      H13, H14, H15               :array[1..max_num_AX,1..max_num_BX] of real;

      x1, x2                      :integer;

begin       {Start of code section for TRANSF}

      matidn(H10,AX,AX,AX,AX);
      for x1 := 1 to AX do
      begin
         for x2 := 1 to AX do
         begin
            H10[x1,x2] := IDEN[x1,x2];
         end;
      end;

      matmin(H10,A,AX,AX,AX,AX);
      for x1 := 1 to AX do
      begin
         for x2 := 1 to AX do
         begin
            H11[x1,x2] := MINU[x1,x2];
         end;
      end;

      matmul(H11,AX,AX,AX,AX);
      for x1 := 1 to AX do
      begin
         for x2 := 1 to AX do
         begin
            H12[x1,x2] := INVS[x1,x2];
         end;
      end;

      matmul(H12,B,AX,AX,AX,BX);
      for x1 := 1 to AX do
      begin
         x2 := 1;
```

```
            begin
               H13[x1,x2] := PROD[x1,x2];
            end;
         end;

         matmul(C,H13,AX,AX,AX,BX);
         for x1 := 1 to AX do
         begin
            x2 := 1;
            begin
               H14[x1,x2] := PROD[x1,x2];
            end;
         end;

         matplu(H14,D,AX,BX,AX,BX);
         for x1 := 1 to AX do
         begin
            x2 := 1;
            begin
               H15[x1,x2] := PLUS[x1,x2];
            end;
         end;

         writeln(lst,'H15');
         writeln(lst,'---');
         matpri(H15,AX,1);
         writeln(lst);

         writeln(lst,'H12');
         writeln(lst,'---');
         matpri(H12,AX,AX);
         writeln(lst);

{Fadeev algorithm}

      twodgraph;

end;    {Of code section of TRANSF}


procedure poles;

{
 Procedure to calculate the poles of the circuit - the roots of
 the characteristic equation det(sI-A) = 0
}

var   num, number                  :integer;

      x1, x2                       :integer;

begin       {Start of code section for POLES}

      matidn(G1,AX,AX,AX,AX);
      for x1 := 1 to AX do
```

```
begin
   for x2 := 1 to AX do
   begin
      G1[x1,x2] := IDEN[x1,x2];
   end;
end;

matmul(A,G1,AX,AX,AX,AX);
for x1 := 1 to AX do
begin
   for x2 := 1 to AX do
   begin
      F1[x1,x2] := PROD[x1,x2];
   end;
end;

matmul(F1,G1,AX,AX,AX,AX);
for x1 := 1 to AX do
begin
   for x2 := 1 to AX do
   begin
      I1[x1,x2] := PROD[x1,x2];
   end;
end;

matmul(A,G1,AX,AX,AX,BX);
for x1 := 1 to AX do
begin
   x2 := 1;
   begin
      I2[x1,x2] := PROD[x1,x2];
   end;
end;

matplu(I1,I2,AX,BX,AX,BX);
for x1 := 1 to AX do
begin
   x2 := 1;
   begin
      G2[x1,x2] := PLUS[x1,x2];
   end;
end;

matmul(A,G2,AX,AX,AX,BX);
for x1 := 1 to AX do
begin
   x2 := 1;
   begin
      F2[x1,x2] := PROD[x1,x2];
   end;
end;

matmul(F2,G1,AX,AX,AX,AX);
for x1 := 1 to AX do
begin
```

```
    for x2 := 1 to AX do
    begin
       I3[x1,x2] := PROD[x1,x2];
    end;
end;

matmul(A,G2,AX,AX,AX,BX);
for x1 := 1 to AX do
begin
    x2 := 1;
    begin
       I4[x1,x2] := PROD[x1,x2];
    end;
end;

matplu(I3,I4,AX,BX,AX,BX);
for x1 := 1 to AX do
begin
    x2 := 1;
    begin
       G3[x1,x2] := PLUS[x1,x2];
    end;
end;

matmul(A,G3,AX,AX,AX,BX);
for x1 := 1 to AX do
begin
    x2 := 1;
    begin
       F3[x1,x2] := PROD[x1,x2];
    end;
end;

matmul(F3,G1,AX,AX,AX,AX);
for x1 := 1 to AX do
begin
    for x2 := 1 to AX do
    begin
       I5[x1,x2] := PROD[x1,x2];
    end;
end;

matmul(A,G3,AX,AX,AX,BX);
for x1 := 1 to AX do
begin
    x2 := 1;
    begin
       I6[x1,x2] := PROD[x1,x2];
    end;
end;

matplu(I5,I6,AX,BX,AX,BX);
for x1 := 1 to AX do
begin
    x2 := 1;
```

```
            begin
               G4[x1,x2] := PLUS[x1,x2];
            end;
         end;

         matmul(A,G4,AX,AX,AX,BX);
         for x1 := 1 to AX do
         begin
            x2 := 1;
            begin
               F3[x1,x2] := PROD[x1,x2];
            end;
         end;

         writeln(lst,'G4');
         writeln(lst,'--');
         writeln(lst,'Poles of system:');
         matpri(G4,AX,AX);
         writeln(lst);

end;    {Of code section of POLES}


procedure zeros;

{
 Procedure to calculate the zeros of the circuit - the roots of
 the characteristic equation adj(sI-A) = 0
}

var    num, number                    :integer;

begin    {Start of code section for ZEROS}

         writeln(lst,'F4');
         writeln(lst,'--');
         writeln(lst,'Zeros of system:');
         matpri(F4,AX,AX);
         writeln(lst);

end;    {Of code section of ZEROS}


procedure bandth;

{
 Procedure to calculate the bandwidth of the circuit:
}

var    num, number                    :integer;

         maximum,
         gainm, phasem                :real;

begin  {Start of code section of BANDTH}
```

```
        writeln('Bandwidth of circuit');
end;    {Of code section of BANDTH}

procedure stable;

{
 Procedure to calculate the stability of the circuit:
}

var   num, number                    :integer;

      para                           :boolean;

begin     {Start of code section of STABLE}

        para := 'false';
        for num = 1 to AX do
        begin
           for number = 1 to AX do
           begin
              if G4[AX,AX] < 0 then
              begin
                 para := true;
                 if para := false then
                    writeln('Circuit is unstable: Pole at ',G4[AX,AX]);
                    exit;
              end;
           end;
           writeln('Circuit is stable: Pole at ',G4[AX,AX]);
        end;

end;      {Of code section of STABLE}


procedure fbresp;

{
 Procedure to calculate the feedback response of the circuit:
}

var   num, number                    :integer;

      minimum, negative,
      stabm                          :real;

begin     {Start of code section of FBRESP}

         writeln('Feedback response of circuit');

end;      {Of code section of FBRESP}
```

```
procedure polesh;

{
 Pole-shifting by state feedback. p. 174
}

var   num, number                    :integer;

begin     {Start of code section of POLESH}

        writeln('Pole-shifting by state feedback');
        {x = (A-BK)x + Bv}

end;      {Of code section of POLESH}


begin     {Start of code section for FREQUENCY}

     {header;  }              {Printout and display values in use}

     for nem := 1 to num2 do
     begin
        if ANLY1[nem,1] = 'TRANSF' then
        begin
           transf;
           for nem1 := 1 to num2 do
           begin
              if ANLY1[nem,1] = 'POLES' then
              begin
                 poles;
              end
              else if ANLY1[nem,1] = 'ZEROS' then
              begin
                 zeros;
              end
              else if ANLY1[nem,1] = 'STABLE' then
              begin
                 stable;
              end;
           end;
        end;
     end;

     for nem := 1 to num2 do
     begin
        if ANLY1[nem,1] = 'BANDTH' then
        begin
           bandth;
        end;
     end;

     for nem := 1 to num2 do
     begin
        if ANLY1[nem,1] = 'FBRESP' then
        begin
```

```
            fbresp;
            for nem1 := 1 to num2 do
            begin
               if ANLY1[nem,1] = 'POLESH' then
               begin
                  polesh;
               end;
            end;
         end;
      end;

end;        {Of code section of FREQUENCY}

END.
```

```
{$B+}   {Boolean complete evaluation on}
{$S+}   {Stack checking on}
{$I+}   {I/O checking on}
{$N+}   {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT GCHEAD;        {Version 1      Created on 7th May 1991}

{$R+}   {Compiler range checking for arrays, etc.}
{$F+}   {Far call compiler directive}
{$O+}   {Overlay compiler directive}


{
 GCHEAD is an overlaid unit of GCAP03, a Generalised Circuit Analysis Program
 for pulsed power.  This is version 1 of the unit which is used to provide
 both an on-screen and hard copy printout of the parameters in use in the
 current simulation.  An additional line on the screen keeps the user
 informed of what's happening when a run is in progress.  GCHEAD contains
 a nested procedure, SCREEN.
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Printer,
  GcInit;


TYPE  textline = string [80];

VAR   number   :          string [10];

procedure header;

Implementation

{-------------------------------------------------------------------------}

{Now declare the procedures and functions}

PROCEDURE HEADER;       {Printout and display values in use}

procedure SCREEN (outstring :textline); {Writes to both screen and printer}

begin           {Start of code section for SCREEN procedure}
   writeln(outstring, lf);
   writeln(lst, outstring, lf);
end;            {End of code section for SCREEN}
```

```
begin               {Start of code section for HEADER procedure}
   clrscr;
      writeln(lst,'=========================================');
      writeln(lst,title);
      writeln(lst,'=========================================',lf);
      writeln(lst);

      str(volt_supply:10, number);
      screen(line['g'] + number);
      str(cap_stor:3:1, number);
      screen(line['h'] + number);
      writeln(lst,'=========================================',lf);
      writeln(lst);
writeln(lf, 'Run in progress with the above values: ');
writeln(lf, 'round(sim_time*1000/inc_time),' steps',lf);

end;          {Of HEADER procedure}

END.
```

```
{$B+}   {Boolean complete evaluation on}
{$S+}   {Stack checking on}
{$I+}   {I/O checking on}
{$N+}   {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT GCOUT1;        {Version 1      Created on 11th June 1991}

{$R+}   {Compiler range checking for arrays, etc.}
{$F+}   {Far call compiler directive}
{$O+}   {Overlay compiler directive}

{
 GCOUT1 is an overlaid unit of GCAP03, a Generalised Circuit Analysis Program
 for pulsed power.  This is version 1 of the unit which provides a tabular
 display of the results of a simulation.
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Graph,
  Printer,
  GcInit;

procedure table;

Implementation

{Start of global type, constant and variable definitions}

VAR   GraphDriver                                   :integer;
      GraphMode                                     :integer;
      ErrorCode                                     :integer;

      Intensity                                     :array[0..30] of real;
      x_axis, y_axis, z_axis                        :string[25];

      step                                          :integer;
      PixelColor,Size                               :word;

CONST Gray50:   FillPatternType     = ($AA, $55, $AA, $55,
                                       $AA, $55, $AA, $55);

{------------------------------------------------------------------------}

{Now declare the procedures and functions}
```

```
PROCEDURE TABLE;

{
 This procedure is used to provide a tabular listing of the parameters
 resulting from a simulation.
}

var   num, nom, number,
      incr, x_step, index                    :integer;

      range                                  :real;

      heading                                :string[30];

      STATE                                  :array[1..100,1..4] of real;
      PARAMETER                              :array[1..100] of real;

begin                    {Start of code section for TABLE}

     writeln(lst,'TABLE');

     GraphDriver := Detect;                          {Set flag: do detection}
     InitGraph(GraphDriver, GraphMode, 'C:\DRIVERS');
     ErrorCode := GraphResult;
     if ErrorCode <> grOk then                       {Is there an error?}
     begin
          writeln('Graphics error: ', GraphErrorMsg(ErrorCode));
          writeln('Program aborted...');
          Halt(1);
     end;

     for num := 1 to num3 do
     begin
        if STRC1[num,1] = 'TABLE' then
        begin
           incr := round(STRC2[num,1]/inc_time);
           range := STRC2[num,2]/inc_time;
        end;
     end;

     for num := 1 to num3 do
     begin
        for nom := 1 to num1 do
        begin
           if STRC1[num,2] = COMP7[nom,2] then
           begin
              for index := 1 to round(range) do
              begin
                 STATE[index,1] := VOLTAGE[nom,index];
                                   {Voltage across a component}
                 STATE[index,2] := CURRENT[nom,index];
                                   {Current through a component}
                 STATE[index,3] := VOLTAGE[nom,index]*CURRENT[nom,index];
                                   {Energy stored/power dissipated
                                    in a component}
```

```
        end;
      end;
   end;
end;

for num := 1 to num3 do
begin
   for nom := 1 to num1 do
   begin
      if STRC1[num,3] = 'V' then
      begin
         for index := 1 to round(range) do
         begin
            PARAMETER[index] := STATE[index,1];
                                {Voltage across a component}
         end;
      end
      else if STRC1[num,3] = 'I' then
      begin
         for index := 1 to round(range) do
         begin
            PARAMETER[index] := STATE[index,2];
                                {Current through a component}
         end;
      end
      else if (STRC1[num,3] = 'E') or (STRC1[num,3] = 'P') then
      begin
         for index := 1 to round(range) do
         begin
            PARAMETER[index] := STATE[index,3];
                                {Energy stored/power dissipated
                                 in a component}
         end;
      end;
   end;
end;

number := 0;
for nom := 1 to round(range) do
begin
   if ((nom div incr) = 0) then
   begin
      number := number + 1;
      INTENSITY[number] := PARAMETER[nom];
   end;
end;

for num := 1 to num2 do
begin
   if ANLY1[num,1] = 'DCTRAN' then
   begin
      heading := 'DC Transfer Curve';
      x_step := incr;
   end;
end;
```

```
        writeln(lst,title);
        writeln(lst,'          ');
        writeln(lst,heading);
        writeln(lst,'          ');
        writeln(lst,'Input     ',sp5,'Output     ');
        writeln(lst,'-----     ',sp5,'------     ');

        x_step := 0;
        for nom := 1 to round(range) do
        begin
           x_step := x_step + 1;
           writeln(lst,x_step:11,sp2,INTENSITY[nom]:11);
        end;
        writeln(lst,'          ');


        Sound(1500);
        Delay(300);
        NoSound;

        Readln;
        CloseGraph;

end;          {End of code section of TABLE}

END.
```

```
{$B+}   {Boolean complete evaluation on}
{$S+}   {Stack checking on}
{$I+}   {I/O checking on}
{$N+}   {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT GCOUT2;        {Version 1      Created on 11th June 1991}

{$R+}   {Compiler range checking for arrays, etc.}
{$F+}   {Far call compiler directive}
{$O+}   {Overlay compiler directive}

{
 GCOUT2 is an overlaid unit of GCAP03, a Generalised Circuit Analysis Program
 for pulsed power.  This is version 1 of the unit which provides a
 two-dimensional graphics display of the results of a simulation.
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Graph,
  Printer,
  GcInit;

procedure twodgraph;

Implementation

{Start of global type, constant and variable definitions}

VAR   GraphDriver                          :integer;
      GraphMode                            :integer;
      ErrorCode                            :integer;
      res, space, indx, indy               :integer;
      x_init, y_init, x_stop, y_stop       :array[0..30,0..30] of integer;
      x, y, xx, yy, y1, startx, starty     :integer;
      Intensity                            :array[0..30] of real;
      Value                                :array[0..637] of integer;

      x_axis, y_axis                       :string[25];

      amp, watt, c, d, e                   :real;
      step                                 :integer;
      PixelColor,Size                      :word;

CONST Gray50:    FillPatternType =  ($AA, $55, $AA, $55,
                                    $AA, $55, $AA, $55);
```

```
{-------------------------------------------------------------------------}

{Now declare the procedures and functions}

PROCEDURE TWODGRAPH;

{
 This procedure is used to provide a two-dimensional graphics display
 of the parameters resulting from a simulation.
}
var   num, nom, number,
      incr, x_step, index                    :integer;

      range                                  :real;

      heading                                :string[30];

      STATE                                  :array[1..100,1..4] of real;
      PARAMETER                              :array[1..100] of real;


begin                    {Start of code section for TWODGRAPH}

    writeln(lst,'2DGRAPH');


    GraphDriver := Detect;                          {Set flag: do detection}
    InitGraph(GraphDriver, GraphMode, 'C:\DRIVERS');
    ErrorCode := GraphResult;
    if ErrorCode <> grOk then                       {Is there an error?}
    begin
       writeln('Graphics error: ', GraphErrorMsg(ErrorCode));
       writeln('Program aborted...');
       Halt(1);
    end;

    for num := 1 to num3 do
    begin
       if STRC1[num,1] = '2DGRAPH' then
       begin
          incr := round(STRC2[num,1]/inc_time);
          range := STRC2[num,2]/inc_time;
       end;
    end;

    for num := 1 to num3 do
    begin
       for nom := 1 to num1 do
       begin
          if STRC1[num,2] = COMP7[nom,2] then
          begin
             for index := 1 to round(range) do
             begin
                STATE[index,1] := VOLTAGE[nom,index];
                                  {Voltage across a component}
```

```
            STATE[index,2] := CURRENT[nom,index];
                              {Current through a component}
            STATE[index,3] := VOLTAGE[nom,index]*CURRENT[nom,index];
                              {Energy stored/power dissipated
                               in a component}
         end;
      end;
   end;
end;

for num := 1 to num3 do
begin
   for nom := 1 to num1 do
   begin
      if STRC1[num,3] = 'V' then
      begin
         for index := 1 to round(range) do
         begin
            PARAMETER[index] := STATE[index,1];
                              {Voltage across a component}
         end;
      end
      else if STRC1[num,3] = 'I' then
      begin
         for index := 1 to round(range) do
         begin
            PARAMETER[index] := STATE[index,2];
                              {Current through a component}
         end;
      end
      else if (STRC1[num,3] = 'E') or (STRC1[num,3] = 'P') then
      begin
         for index := 1 to round(range) do
         begin
            PARAMETER[index] := STATE[index,3];
                              {Energy stored/power dissipated
                               in a component}
         end;
      end;
   end;
end;

number := 0;
for nom := 1 to round(range) do
begin
   if ((nom div incr) = 0) then
   begin
      number := number + 1;
      INTENSITY[number] := PARAMETER[nom];
   end;
end;

for num := 1 to num2 do
begin
   if ANLY1[num,1] = 'DCTRAN' then
```

```
      begin
         heading := 'DC Transfer Curve';
         y_axis := 'Output voltage (V)';
         x_axis := 'Input voltage (V)';
         x_step := incr;
      end
      else if ANLY1[num,1] = 'ACTRAN' then
      begin
         heading := 'AC Transfer Curve';
         y_axis := 'Output voltage (V)';
         x_axis := 'Input voltage (V)';
         x_step := incr;
      end
      else if ANLY1[num,1] = 'RESPON' then
      begin
         heading := 'Total Response';
         y_axis := 'Output voltage (V)';
         x_axis := 'Time (ns)';
         x_step := incr;
      end
      else if ANLY1[num,1] = 'BODE' then
      begin
         heading := 'Bode Plot';
         y_axis := 'Transfer Function (dB)';
         x_axis := 'Frequency (Hz)';
         x_step := incr;
      end;
   end;

   writeln(lst,title);
   writeln(lst,'        ');
   writeln(lst,heading);
   writeln(lst,'        ');

   x_step := 0;
   for nom := 1 to round(range) do
   begin
      x_step := x_step + 1;
      writeln(lst,x_step:11,sp2,INTENSITY[nom]:11);
   end;
   writeln(lst,'        ');


    for space := 30 downto 1 do
    begin
       SetColor(5);
       for res := 1 to 30 do
       begin
          x_init[res,space] := 250-(5*(space-1))+(10*(res-1));
          y_init[res,space] := 400-(5*(space-1))-round(intensity[space]);
             startx := x_init[1,space];
             starty := y_init[1,space];
          x_stop[res,space] := 250-(5*(space-1))+(10*(res));
          y_stop[res,space] := 400-(5*(space-1))-round(intensity[space]);
Line(x_init[res,space], y_init[res,space], x_stop[res,space], y_stop[res,space]);
```

```
          if space < 30 then
Line(x_stop[res,space], y_stop[res,space], x_stop[res,space+1], y_stop[res,space+1]);
       end;


  for xx := startx to x_stop[res,space] do
  begin
     for yy := 1 to starty do
     begin
        PixelColor := GetPixel(xx,yy);
        if PixelColor = 5 then
        begin
           y1 := yy;
           PutPixel(xx,y1,15);
        end;
     end;

     for yy := (y1+1) to GetMaxY do
     PutPixel(xx,yy,0);

  end;

  end;

     SetColor(3);
     Line(100, 50, 100, 250);                    { DRAWS y AXIS LINE }
     Line(250, 400, 600, 400);                   { DRAWS x AXIS LINE }
     Rectangle(0, 0, GetMaxX, GetMaxY);          {Draw full screen box}
     SetTextJustify(CenterText, CenterText);     {Center text}
     SetTextStyle(SansSerifFont, VertDir, 1);
     OutTextXY(30, 140, y_axis);
     SetTextStyle(SansSerifFont, HorizDir, 1);
     OutTextXY(400, 460, x_axis);

     SetTextStyle(SansSerifFont, HorizDir, 1);
     OutTextXY(350, 60, title);

       c := 250/step;
       d := amp/c;

       x := 250;
       While x <= 600 do
       begin                                     {Draws 'I' on x axis }
          Line(x, 400, x, 410);
          x := x+step
       end;

       y :=42;
       While y <= 242 do
       Begin                                     {Draws '_' on y axis }
       OutTextXY(95, y, '_');
       y := y+step
       end;

     Sound(1500);
```

```
        Delay(300);
        NoSound;

        Readln;
        CloseGraph;

end;            {End of code section of TWODGRAPH}

END.
```

```
{$B+}   {Boolean complete evaluation on}
{$S+}   {Stack checking on}
{$I+}   {I/O checking on}
{$N+}   {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT GCOUT3;        {Version 1      Created on 11th June 1991}

{$R+}   {Compiler range checking for arrays, etc.}
{$F+}   {Far call compiler directive}
{$O+}   {Overlay compiler directive}

{
 GCOUT3 is an overlaid unit of GCAP03, a Generalised Circuit Analysis Program
 for pulsed power.  This is version 1 of the unit which provides a
 three-dimensional graphics display of the results of a simulation.
 }

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Graph,
  Printer,
  GcInit;

procedure threedgraph;

Implementation

{Start of global type, constant and variable definitions}

VAR   GraphDriver                          :integer;
      GraphMode                            :integer;
      ErrorCode                            :integer;
      res, space, indx, indy               :integer;
      x_init, y_init, x_stop, y_stop       :array[0..30,0..30] of integer;
      x, y, xx, yy, y1, startx, starty     :integer;
      Intensity                            :array[0..30,0..30] of real;
      Value                                :array[0..637] of integer;
      title                                :string[55];
      x_axis, y_axis, z_axis               :string[25];

      amp, watt, c, d, e                   :real;
      step                                 :integer;
      PixelColor,Size                      :word;

CONST Gray50:   FillPatternType =  ($AA, $55, $AA, $55,
                                    $AA, $55, $AA, $55);
```

```
{-------------------------------------------------------------------------}

{Now declare the procedures and functions}


PROCEDURE THREEDGRAPH;

{
 This procedure is used to provide a three-dimensional graphics display
 of the parameters resulting from a simulation.
}

var   num, nem, nom, number,
      incr, x_step, index                    :integer;

      range                                  :real;

      heading                                :string[30];

      STATE                                  :array[1..100,1..4] of real;
      PARAMETER                              :array[1..100] of real;

begin                    {Start of code section for THREEDGRAPH}

    writeln(lst,'3DGRAPH');

    GraphDriver := Detect;                          {Set flag: do detection}
    InitGraph(GraphDriver, GraphMode, 'C:\DRIVERS');
    ErrorCode := GraphResult;
    if ErrorCode <> grOk then                       {Is there an error?}
    begin
       writeln('Graphics error: ', GraphErrorMsg(ErrorCode));
       writeln('Program aborted...');
       Halt(1);
    end;

    for num := 1 to num3 do
    begin
       if STRC1[num,1] = '3DGRAPH' then
       begin
          incr := round(STRC2[num,1]/inc_time);
          range := STRC2[num,2]/inc_time;
       end;
    end;

    for num := 1 to num3 do
    begin
       for nom := 1 to num1 do
       begin
          if STRC1[num,2] = COMP7[nom,2] then
          begin
             for index := 1 to round(range) do
             begin
                STATE[index,1] := VOLTAGE[nom,index];
                                        {Voltage across a component}
```

```
            STATE[index,2] := CURRENT[nom,index];
                                {Current through a component}
            STATE[index,3] := VOLTAGE[nom,index]*CURRENT[nom,index];
                                {Energy stored/power dissipated
                                 in a component}
         end;
      end;
   end;
end;

for num := 1 to num3 do
begin
   for nom := 1 to num1 do
   begin
      if STRC1[num,3] = 'V' then
      begin
         for index := 1 to round(range) do
         begin
            PARAMETER[index] := STATE[index,1];
                                {Voltage across a component}
         end;
      end
      else if STRC1[num,3] = 'I' then
      begin
         for index := 1 to round(range) do
         begin
            PARAMETER[index] := STATE[index,2];
                                {Current through a component}
         end;
      end
      else if (STRC1[num,3] = 'E') or (STRC1[num,3] = 'P') then
      begin
         for index := 1 to round(range) do
         begin
            PARAMETER[index] := STATE[index,3];
                                {Energy stored/power dissipated
                                 in a component}
         end;
      end;
   end;
end;

number := 0;
for nom := 1 to round(range) do
begin
   for nem := 1 to round(range) do
   begin
      if ((nom div incr) = 0) then
      begin
         number := number + 1;
         INTENSITY[number,number] := PARAMETER[nom];
      end;
   end;
end;
```

```
      for num := 1 to num2 do
      begin
         if ANLY1[num,1] = 'DCTRAN' then
         begin
            heading := 'DC Transfer Curve';
            y_axis := 'Output voltage (V)';
            x_axis := 'Input voltage (V)';
            x_step := incr;
         end
         else if ANLY1[num,1] = 'ACTRAN' then
         begin
            heading := 'AC Transfer Curve';
            y_axis := 'Output voltage (V)';
            x_axis := 'Input voltage (V)';
            x_step := incr;
         end
         else if ANLY1[num,1] = 'RESPON' then
         begin
            heading := 'Total Response';
            y_axis := 'Output voltage (V)';
            x_axis := 'Time (ns)';
            x_step := incr;
         end
         else if ANLY1[num,1] = 'BODE' then
         begin
            heading := 'Bode Plot';
            y_axis := 'Transfer Function (dB)';
            x_axis := 'Frequency (Hz)';
            x_step := incr;
         end;
      end;

      writeln(lst,title);
      writeln(lst,'        ');
      writeln(lst,heading);
      writeln(lst,'        ');

      x_step := 0;
      for nom := 1 to round(range) do
      begin
         for nem := 1 to round(range) do
         begin
            x_step := x_step + 1;
            writeln(lst,x_step:11,sp2,INTENSITY[nom,nem]:11);
         end;
      end;
      writeln(lst,'        ');

for indx := 29 downto 15 do
begin
   for indy := 1 to 30 do
   begin
      intensity[indy,indx] := intensity[indy,indx+1] - (28-indx);
   end;
end;
```

```
for indx := 14 downto 0 do
begin
   for indy := 1 to 30 do
   begin
      intensity[indy,indx] := intensity[indy,indx+1] + (indx);
   end;
end;


      for space := 30 downto 1 do
      begin
         SetColor(5);
         for res := 1 to 30 do
         begin
            x_init[res,space] := 250-(5*(space-1))+(10*(res-1));
            y_init[res,space] := 400-(5*(space-1))-round(intensity[res-1,space]);
               startx := x_init[1,space];
               starty := y_init[1,space];
            x_stop[res,space] := 250-(5*(space-1))+(10*(res));
            y_stop[res,space] := 400-(5*(space-1))-round(intensity[res,space]);
Line(x_init[res,space], y_init[res,space], x_stop[res,space], y_stop[res,space]);
            if space < 30 then
Line(x_stop[res,space], y_stop[res,space], x_stop[res,space+1], y_stop[res,space+1]);
       end;

                                                {FILLS IN BACK LINES - EXCEPT    }
  for xx := startx to x_stop[res,space] do
  begin
     for yy := 1 to starty do
     begin
        PixelColor := GetPixel(xx,yy);
        if PixelColor = 5 then
        begin
           y1 := yy;
           PutPixel(xx,y1,15);
        end;
     end;

     for yy := (y1+1) to GetMaxY do
     PutPixel(xx,yy,0);

  end;

  end;

     SetColor(3);
     Line(100, 50, 100, 250);                   { DRAWS y AXIS LINE }
     Line(100, 250, 250, 400);                  { DRAWS z AXIS LINE }
     Line(250, 400, 600, 400);                  { DRAWS x AXIS LINE }
     Rectangle(0, 0, GetMaxX, GetMaxY);         {Draw full screen box}
     SetTextJustify(CenterText, CenterText);    {Center text}
     SetTextStyle(SansSerifFont, VertDir, 1);
     OutTextXY(30, 140, y_axis);
     SetTextStyle(SansSerifFont, HorizDir, 1);
```

```
        OutTextXY(400, 460, x_axis);
        SetTextStyle(SansSerifFont, HorizDir, 1);
        OutTextXY(70, 330, z_axis);

        SetTextStyle(SansSerifFont, HorizDir, 1);
        OutTextXY(350, 60, title);


        SetFillStyle(9,15);          {interleaving white line fill}
        FloodFill(350,350,15);       {seed point within boundary to be filled}


          c := 250/step;
          d := amp/c;

          x := 250;
          While x <= 600 do
          begin                              {Draws 'I' on x axis }
          Line(x, 400, x, 410);
          x := x+step
          end;

          y :=42;
          While y <= 242 do
          Begin                              {Draws '_' on y axis }
          OutTextXY(95, y, '_');
          y := y+step
          end;

          x :=95;
          y :=242;
          While (x <= 250) and (y <= 400) do
          Begin                              {Draws '_' on z axis }
          OutTextXY(x, y, '_');
          y := y+step;
          x := x+step
          end;

          Sound(1500);
          Delay(300);
          NoSound;

        Readln;
        CloseGraph;

end;        {End of code section of THREEDGRAPH}

END.
```

```
{$B+}   {Boolean complete evaluation on}
{$S+}   {Stack checking on}
{$I+}   {I/O checking on}
{$N+}   {Numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}


UNIT GCEQNS;        {Version 1      Created on 7th May 1991}

{$R+}   {Compiler range checking for arrays, etc.}
{$F+}   {Far call compiler directive}
{$O+}   {Overlay compiler directive}

{
 GCEQNS is an overlaid unit of GCAP03, a Generalised Circuit Analysis Program
 for pulsed power.  This is version 1 of the unit which is lists the state
 variables and circuit equations required by GCAP.
}

{Start of global type, constant and variable declarations}

Interface

Uses
  Overlay,
  Crt,
  Printer,
  GcInit;

VAR       RX                                        :integer;

procedure state_variables;
procedure equations;

Implementation

{-------------------------------------------------------------------------}

{Now declare the procedures and functions}

PROCEDURE STATE_VARIABLES;

var  result                                         :real;

begin            {Start of code section for STATE_VARIABLES procedure}

     writeln(lst, title, cr);

{State variables considered in the model}

{
     SYMBOL   PARAMETER
     ------   ---------
     Vc       Capacitor voltage
```

```
        qc          Capacitor charge

        iL          Inductor current
        fL          Inductor flux

        V           Switch voltage
        i           Switch current
}

end;    {Of code section of STATE_VARIABLES}

PROCEDURE EQUATIONS;

var     result                                              :real;

begin           {Start of code section for EQUATIONS procedure}

    writeln(lst, title, cr);

{
    Equation No. 1 is   x = Ax + Bu             STATE EQUATION

    Equation No. 2 is   y = Cx + Du + Eu        INPUT-STATE-OUTPUT
                                                EQUATION

    Equation No. 3 is   x(t) = f(x(t),u(t),t)   NON-LINEAR
                                                STATE EQUATION

    Equation No. 4 is   y(t) = g(x(t),u(t),t)   NON-LINEAR
                                                INPUT-STATE-OUTPUT
                                                EQUATION

    Equation No. 5 is   x(t) = (Vc(t),iL(t))    STATE VECTOR

    Equation No. 6 is   qc(t) = C(t)Vc(t)       STATE EQUATIONS
                        fL(t) = L(t)iL(t)

    Equation No. 7 is   x(t) = (qc(t),fL(t))    NON-LINEAR
                                                STATE VECTOR

    Equation No. 8 is   i(t) = C(t)Vc(t) + C(t)Vc(t)
                        V(t) = L(t)iL(t) + L(t)iL(t)
                                                NON-LINEAR
                                                STATE EQUATIONS

    Equation No. 9 is   ZIR(t) = exp(At)x(to)   ZERO INPUT RESPONSE

    Equation No.10 is   ZSR(t) = exp(At)B(t)u(t)dt
                                                ZERO STATE RESPONSE

    Equation No.11 is   H(s) = C(sI-A)-1 + D    TRANSFER FUNCTION

    Equation No.12 is   H(s) = -CA-1B + D       DC TRANSFER FUNCTION

    Equation No.13 is   det(sI-A) = 0           SYSTEM POLES
```

Equation No.14 is adj(sI-A) = 0 SYSTEM ZEROS

Equation No.15 is x = (A-BK)x + Bu' CLOSED-LOOP SYSTEM RESPONSE

)

end; {Of code section EQUATIONS}

END.